Optimisation heuristique et génétique de visualisations 2D et 3D dans OLAP : premiers résultats

Florian Sureau*, Fatma Bouali**,*, Gilles Venturini*

* Université François-Rabelais de Tours, Laboratoire d'Informatique
64 avenue Jean Portalis, 37200 Tours, France
venturini@univ-tours.fr
http://www.antsearch.univ-tours.fr
** Université de Lille2, IUT, Dpt STID
25-27 Rue du Maréchal Foch, 59100 Roubaix, France
Fatma.Bouali@univ-lille2.fr

Résumé. Nous étudions dans cet article comment réorganiser les données dans une analyse OLAP afin d'améliorer la visualisation présentée aux décideurs. Après un état de l'art sur la problématique de la réorganisation de matrices et de cubes OLAP, nous présentons deux méthodes. La première est une méthode heuristique permettant de replacer les modalités des dimensions en maximisant la similarité entre deux modalités voisines. La deuxième méthode est un algorithme génétique permettant de faire évoluer le cube de données afin de maximiser un critère d'évaluation de la visualisation. Nous comparons ces deux méthodes avec d'autres approches sur des bases réelles afin d'optimiser la visualisation de matrices (2D) et de cubes (3D) et en présentant les visualisations obtenues. Nous détaillons l'intégration de ces méthodes dans un serveur OLAP ainsi qu'une première ébauche de visualisation obtenue en réalité virtuelle.

1 Introduction

OLAP est un ensemble d'outils permettant de faire une analyse interactive de données sur de multiples dimensions divisées en modalités. Ces données sont représentées sous la forme d'un cube (ou hypercube) dont chaque cellule représente une mesure au croisement de chaque modalité. Plusieurs opérateurs permettent de manipuler l'hypercube afin de sélectionner les dimensions et la granularité de ces données pour les adapter à la demande et aux besoins d'informations de l'utilisateur. Aujourd'hui, l'outil d'analyse le plus classique est le tableau croisé ou le tableau de bord. Un certain nombre de travaux se sont intéressés à la manière d'améliorer les visualisations OLAP et les interactions avec l'utilisateur. Ainsi DBMiner Han et al. (1996), Dive-On Ammoura et al. (2001) ou encore Diva Bulusu (2003) proposent une visualisation sur trois dimensions et pour les deux derniers travaux une première approche en réalité virtuelle. Ces travaux cherchent donc bien à améliorer l'interface entre OLAP et l'expert du domaine en proposant de nouvelles visualisations mais ils ne cherchent pas à les réorganiser pour les rendre plus lisibles et plus informatives. Pourtant, l'aspect décisionnel d'OLAP réside

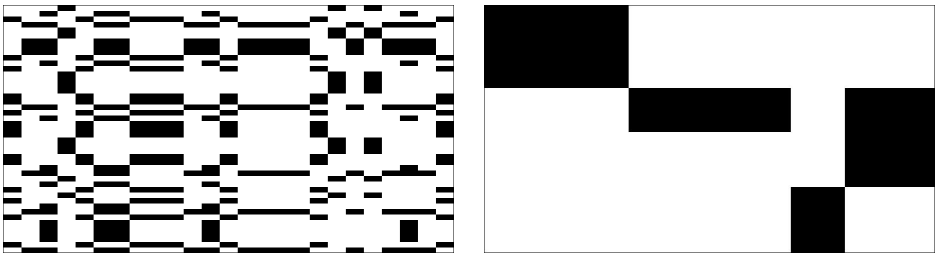


FIG. 1 – *Exemple théorique de réorganisation d'une matrice de données inspiré de Caraux (1984).*

justement dans la découverte de visualisations pertinentes pour l'utilisateur et le problème de fournir des visualisations facilement exploitables se pose de manière cruciale.

Notre problématique concerne donc la mise en oeuvre de méthodes améliorant la qualité "visuelle" et informationnelle d'un cube OLAP. Notre contribution principale dans cet article est de proposer une réorganisation automatique de l'ordre des modalités des dimensions, et une deuxième contribution qui est au tout début de son développement est d'utiliser un environnement de réalité virtuelle avec une perception stéréoscopique pour visualiser et explorer ce cube.

La suite de cet article est organisée comme suit : la section 2 précise le contexte et les études similaires réalisées principalement dans la réorganisation de dimensions. La section 3 décrit les approches heuristiques que nous avons conçues et qui vont optimiser chaque dimension de la visualisation pour la rendre plus lisible. La section 4 décrit l'algorithme génétique que nous avons testé sur cette approche et qui utilise des opérateurs fondés sur les permutations de valeurs avec une optimisation plus globale que la précédente. La section 5 présente les résultats de ces algorithmes appliqués sur trois bases de données ainsi que la comparaison de ces méthodes par rapport à des travaux classiques ainsi que des travaux récents. Nous concluons ce travail dans la section 6 en présentant des perspectives.

2 Problématique et état de l'art

2.1 Formalisation du problème

Nous nous intéressons à la réorganisation des données dans le cube de façon à améliorer la visualisation. En effet, si les données intéressantes (et similaires) sont proches les unes des autres, l'utilisateur aura une analyse beaucoup plus rapide et intuitive. A titre d'exemple, la figure 1 inspirée de Caraux (1984) montre une matrice avant et après réorganisation (permutations) des lignes et colonnes. Elle souligne très bien le fait que le regroupement des données est important en termes de compréhension pour l'utilisateur car en regroupant les données par zones, on peut définir des tendances de valeurs et former des groupes de modalités dont les valeurs de la mesure se ressemblent.

Nous considérons donc un cube de données C et particulièrement l'ensemble de ses dimensions dont les modalités peuvent être réordonnées : il se peut par exemple que l'utilisa-

teur ne souhaite pas réorganiser une dimension de type temporel. Nous notons l'ensemble des dimensions $\mathcal{DIM} = \{D_i\}_{\{1 \leq i \leq n\}}$. Pour chaque dimension D_i nous notons son ensemble de modalités $\mathcal{M}^i$ de taille m^i par $\mathcal{M}^i = \{M^i_j\}_{\{1 \leq j \leq m^i\}}$. Chaque croisement des modalités contient une mesure notée $V(M^1_{i1}, ..., M^d_{in})$.

Notre but est donc de réaliser une permutation des modalités sur chaque dimension afin de maximiser la lisibilité de la visualisation. Nous notons $D = \{D_1, ..., D_d\}$ le sous ensemble de dimensions que nous souhaitons réorganiser (les autres dimensions étant donc $\{D_{d+1}, ..., D_n\}$). Sachant que le nombre d'arrangements est $m^1! \times ... \times m^d!$, il n'est pas possible d'en tester la totalité. C'est pourquoi nous nous intéressons ici à des algorithmes approximatifs pouvant donner une solution correcte en un temps raisonnable. Nous verrons par la suite que les algorithmes peuvent êtres appliqués lors de la visualisation ou plutôt lors de la génération de l'hypercube suivant le temps de traitement demandé. Cet aspect temporel est d'autant plus important que dans OLAP, l'analyse des données a lieu bien souvent de manière interactive avec l'utilisateur.

Enfin, une partie cruciale dans la définition du problème consiste à définir une fonction d'évaluation de la lisibilité d'un cube. Cette fonction doit permettre de décider si un arrangement particulier des données est meilleur qu'un autre du point de vue de l'expert (voir section 4.2).

2.2 Approches existantes

Nous allons montrer que le problème général qui nous intéresse est étudié depuis longtemps et a fait l'objet de nombreuses publications dans des domaines variés (voir par exemple les méthodes mentionnées dans Climer et Zhang (2006) auxquelles on peut notamment ajouter les références citées dans cette section) mais très peu dans OLAP à notre connaissance. Historiquement, la réorganisation des données dans un tableau trouve par exemple ses racines dans des travaux comme Kulezynski (1927) en phytosociologie.

Dans le domaine de la recherche opérationnelle, on peut citer différentes références comme par exemple le problème de localisation optimale de composants d'ordinateurs dans Hanan et Kurtzberg (1972). Des méthodes s'intéressent également à la diagonalisation de matrices pour structurer les données autour de la diagonale telle que Périn et Legoux (1980) par une méthode de "permutation matricielle automatique". En gestion de production, la technologie de groupe de Kusiak (1987) permet d'affecter des travaux à des machines selon la diagonale d'une matrice. Ces techniques tentent donc de regrouper les valeurs des modalités autour de la diagonale ce qui n'est pas complètement adapté à notre problématique puisque l'éparsité de notre cube de données peut être très différente d'une base à une autre et notre but premier est de regrouper au maximum les valeurs similaires et non pas de faire de l'affectation de tâches. Un des algorithmes classiques pour la réorganisation est BEA ("Bond Energy Algorithm" McCormick et al. (1972), nous en présentons des résultats dans la section 5). Enfin, des auteurs ont utilisé directement des "solver" dédiés au problème du voyageur de commerce (PVC) pour optimiser indépendamment les dimensions de matrices (Lenstra (1974)). En effet, trouver une permutation pour une dimension revient à résoudre un PVC (une modalité est une ville, la distance entre modalités est leur similarité). Cette méthode donne de bons résultats notamment pour des problèmes avec un grand nombre de modalités par dimension, mais elle est cependant limitée

à une optimisation indépendante de chaque modalité (voir section 3) plutôt que globale (voir section 4).

Dans Caraux (1984) sont décrits les problèmes rencontrés dans la réorganisation de matrices suivant les lignes et colonnes, et un panel des solutions proposées. Dans Bertin (1977) on s'intéresse à la visualisation de données matricielles et à la réorganisation interactive d'une matrice. Enfin, le problème du bi-partitionnement en classification non supervisée peut aussi être relié à notre problématique, et on peut citer les travaux de Robardet (2002).

Un certain nombre de travaux sont également plus proches de notre problématique et d'OLAP. On trouve par exemple dans le domaine de la compression ou de l'approximation de cubes de données des travaux dans lesquels les auteurs s'intéressent à réduire l'espace de données afin de minimiser l'espace de représentation Sismanis et al. (2002). Dans cet article, on cherche à réduire l'espace de stockage du cube afin de supprimer des redondances dans les données et ainsi de simplifier la visualisation. Cependant, ces travaux sont spécifiques au domaine de la réduction de la place de stockage ou du nombre de requêtes d'agrégations et ne sont donc pas totalement adaptés à notre problème plutôt centré sur la visualisation. Les travaux qui semblent les plus proches sont ceux de Choong et al. (2003) où il est proposé de réorganiser le cube de données en définissant un ordre d'arrangement pour évaluer la qualité de la position des cellules. Le critère de qualité de la visualisation tend à placer les valeurs élevées dans un coin et les plus petites dans le coin opposé. Egalement, dans Ben Messaoud (2006) on utilise une analyse en composantes multiples (ACM, Benzécri (1973)) afin de réduire l'éparsité du cube, c'est à dire tenter de rapprocher les valeurs le plus possible pour ne pas laisser de données seules loin des autres.

Nous poursuivons le même objectif que ces dernières références mais en utilisant de nouveaux algorithmes de réorganisation et en effectuant une comparaison entre plusieurs méthodes sur des données réelles notamment.

3 Algorithmes heuristiques

3.1 Méthodes proposées

Les heuristiques que nous proposons vont optimiser chaque dimension indépendamment les unes des autres et donc sans aucune sensibilité à l'ordre de traitement de ces dimensions. Elles réalisent donc une optimisation locale en appliquant les mêmes opérations à chaque dimension, et nous simplifions donc la discussion dans cette section en parlant d'une dimension seulement que nous notons D_i. Egalement, nous exprimons les complexités de calcul en nombre de distances calculées entre modalités (ou d'opérations de complexité similaire).

Le première heuristique possible (notée H_{tri}) consiste simplement à trier les modalités selon la somme des mesures sur les autres dimensions du cube ($n-1$). Dans le cas d'une mesure à valeurs positives, elle placerait les modalités les plus "vides" au début et les plus "remplies" à la fin de l'axe de visualisation. Cette heuristique est déterministe et a pour complexité $O(m^i log(m^i))$.

La deuxième heuristique H_{Extrem} est la suivante : elle part d'une liste L de modalités qui est vide initialement. Elle commence par sélectionner les deux modalités de D_i les plus proches (voir la distance décrite dans la section suivante) et elle les ajoute à L. Ensuite, elle choisit parmi les modalités de D_i restantes celle qui est la plus proche du premier ou du der-

nier élément de L, et elle insère dans L la modalité à l'extrémité ainsi déterminée. Une fois que toutes les modalités sont placées, D_i est réorganisée suivant l'ordre indiqué par L. Cette heuristique est à nouveau déterministe et donne toujours le même résultat pour un cube donné. Sa complexité est en $O(m^{i^2})$.

Ensuite, nous proposons de tester des heuristiques qui vont insérer des modalités dans tous les emplacements possible de L. L'algorithme servant de base à ce principe est le suivant : nous plaçons dans L les deux modalités de D_i les plus éloignées, puis nous choisissons une troisième modalité la plus dissimilaire possible des deux premières (minimisant la somme des deux similarités), et nous l'insérons entre les deux premières. A partir de cette initialisation de L, nous insérons les autres modalités en considérant que la liste L est toroïdale (le premier et le dernier élément sont voisins). Pour chaque modalité m restant à placer, nous considérons chaque couple de modalités adjacentes (m_{gauche}, m_{droite}) dans L et nous insérons m à l'emplacement qui minimise la fonction *Stratégie*($m_{gauche}, m, m_{droite}$) qui peut être l'une des suivantes :

- Max($dist(m_{gauche}, m), dist(m, m_{droite})$) : minimise la plus grande distance à gauche ou à droite,
- Min($dist(m_{gauche}, m), dist(m, m_{droite})$) : minimise la plus petite distance,
- Moyenne($dist(m_{gauche}, m), dist(m, m_{droite})$) : minimise la moyenne.

Nous les notons respectivement $H_{InsertMax}$, $H_{InsertMin}$ et $H_{InsertMoy}$. Elles sont déterministes et leur complexité est en $O(m^{i^2})$. Nous comparons ces heuristiques avec BEA, reconnue pour son efficacité sur le réarrangement de matrices, dans la section 5.

3.2 Distance entre modalités

Les algorithmes heuristiques que nous proposons se basent sur le calcul d'une distance entre les modalités d'une dimension afin de les replacer au mieux. Pour évaluer la distance entre deux modalités M_j^i et M_k^i d'une dimension D^i, nous calculons la distance euclidienne entre les deux cubes de dimension $n-1$ en fixant respectivement les valeurs de D^i à M_j^i et M_k^i :

$$dist(M_j^i, M_k^i) = \sqrt{\sum[(V(., M_j^i, .) - V(., M_k^i, .)]^2}$$

Dans le cas d'une matrice ($n = 2$), il s'agit d'une distance entre deux vecteurs, et dans le cas d'un cube ($n = 3$) d'une distance entre deux matrices. La distance entre deux modalités sera d'autant plus élevée qu'elles auront des valeurs de mesure différentes sur les autres $n-1$ dimensions considérées. Eventuellement, l'utilisateur peut spécifier les dimensions sur lesquelles est calculée la distance (mais ici par simplification nous considérons l'ensemble des $n-1$ dimensions). Nous abordons dans la section 5.4 un traitement très basique des valeurs $NULL$.

L'algorithme BEA McCormick et al. (1972) se base sur une fonction différente, ME ("Measure of Effectiveness"), pour le calcul de la similarité entre modalités :

$$dist(M_j^i, M_k^i) = \sum[(V(., M_j^i, .) * V(., M_k^i, .)]$$

4 Algorithme génétique

4.1 Principes de l'algorithme

Nous avons voulu testé également une métaheuristique afin de connaître ses performances sur ce problème. Cette métaheuristique va optimiser toutes les dimensions en même temps et réaliser ainsi une optimisation globale (par opposition à l'optimisation locale réalisée par les heuristiques précédentes). Nous avons choisi les algorithmes génétiques (AG, Holland (1975)). Un individu de la population (notée Pop) représente un cube de données et particulièrement un ordre des modalités de chacune des dimensions de D.

La première population est créée en engendrant donc $|Pop|$ individus par permutation aléatoire des modalités du cube initial. Ensuite, nous évaluons chacun des individus avec une fonction H_g qui mesure l'homogénéité du cube (décrite dans la section suivante). Nous générons un individu de la manière suivante : nous sélectionnons deux individus parents par tournoi binaire puis nous créons un enfant E par croisement avec une probabilité de 0.5, et sinon par mutation. E est inséré dans la population à la place du plus mauvais individu si E est meilleur que ce dernier.

L'opérateur de croisement utilisé doit opérer sur des permutations. Nous avons donc choisi un opérateur parmi ceux utilisés dans le problème du PVC, à savoir le PMX (Partially Mapped Crossover, Goldberg (1989)), car celui-ci est connu pour faire hériter des sous-séquences entières. Pour notre problème, cet opérateur semble mieux adapté car il a tendance à respecter les relations de voisinage entre modalités (par opposition à un autre opérateur, l'OX pour Order-based crossover qui lui respecte plutôt l'ordre dans la séquence).

Nous utilisons un opérateur de mutation qui effectue une ascension locale, un "city-swap" amélioré (à l'instar d'un autre opérateur connu pour le PVC appelé "2-opt" et qui inverse la meilleure sous-séquence dans la permutation) : une dimension est sélectionnée au hasard ainsi qu'une modalité de cette dimension. Cette modalité est échangée avec la modalité qui permet de maximiser l'homogénéité du cube (voir section suivante).

Dans nos tests, nous utilisons cet AG en ayant fixé expérimentalement $|Pop| = 30$, et les résultats ont été calculés en moyenne sur 10 essais. L'algorithme est stoppé quand l'homogénéité n'augmente plus sur un certain intervalle de temps, pour éviter un temps de traitement excessivement long pour un gain trop faible.

Base Jouet

Algorithme	Homogénéité	Gain d'homogénéité	Temps de calcul
$H_{initiale}$	8.8%	0%	-
H_{tri}	24.11%	174.6%	172 ms
H_{Extrem}	**24.93**%	**183.9**%	172 ms
H_{BEA}	24.90%	182.9%	172 ms
$H_{InsertMax}$	21.5%	144.87%	172 ms
$H_{InsertMin}$	21.5%	167.65%	172 ms
$H_{InsertMoy}$	24.6%	180.18%	172 ms
AG	20.72%	135.41%	15 min

Base Census-Income

Algorithme	Homogénéité	Gain d'homogénéité	Temps de calcul
$H_{initiale}$	14.16%	0%	-
H_{tri}	20.13%	42.16%	152 ms
H_{Extrem}	20.8%	48.89%	156 ms
H_{BEA}	20.5%	44.77%	149 ms
$H_{InsertMax}$	19.3%	36.29%	152 ms
$H_{InsertMin}$	20.41%	44.13%	156 ms
$H_{InsertMoy}$	19.24%	35.87%	157 ms
AG	**22.63%**	**59.12%**	14 min

Base Mammographies

Algorithme	Homogénéité	Gain d'homogénéité	Temps de calcul
$H_{initiale}$	0.47%	0%	-
H_{tri}	0.54%	14,89%	1.9 s
H_{Extrem}	0.51 %	8,5%	2.5 s
H_{BEA}	**0.57** %	**21,27%**	2.5 s
$H_{InsertMax}$	0.34%	-27,65%	2.4 s
$H_{InsertMin}$	0.39%	-17,02%	2.4 s
$H_{InsertMoy}$	0.35%	-25,53%	2.4 ms
AG	0.47%	0%	40 min

TAB. 1 – *Résultats obtenus sur les trois bases testées par l'ensemble de nos méthodes.*

Bases	n	Modalités/dimension
Jouet	2	45/25
Census-Income	2	23/43
Mammographies	3	173/62/5

TAB. 2 – *Bases de données testées : n est le nombre de dimensions.*

4.2 Fonction d'évaluation

La fonction d'évaluation des individus est celle utilisée dans Ben Messaoud (2006). Pour chaque cellule du cube, on définit localement une similarité avec ses huit voisines (pour une matrice, et 26 voisines pour un cube). Cette similarité est la distance entre la mesure de la cellule et les mesures de ses voisines. On définit ensuite l'homogénéité brute du cube par la somme de ces similarités locales. L'indice d'homogénéité H_g est donc défini par le rapport de l'homogénéité brute sur l'homogénéité maximum, c'est-à-dire un cube de mêmes dimensions contenant des mesures toutes identiques. H_g est donc une approximation pour évaluer la qualité de la représentation en termes de ressemblance des cellules adjacentes, ce qui est un bon indice (qui de plus permet des comparaisons avec des travaux précédents). Néanmoins, un seul indice ne peut sans doute pas à lui seul d'évaluer l'intérêt d'une visualisation pour l'expert, et trouver d'autres indices possibles fait partie de nos perspectives (voir la section 6).

5 Résultats

5.1 Bases de test et méthodologie

Nous avons testé nos algorithmes sur trois bases de données différentes (voir tableau 2) : la première est une base de données jouet nous permettant de tester la qualité de la représentation et de la réorganisation par rapport à la matrice cible (inspirée de Caraux (1984)). Nous considérons ensuite la base de données "Census-Income" [1] utilisée notamment dans les travaux de Ben Messaoud (2006) pour comparer la qualité de la représentation obtenue ainsi que le temps d'exécution par rapport à ces travaux récents. La dernière base de données, également utilisée dans Ben Messaoud (2006), représente des informations sur des images médicales mammographiques et nous permet de tester nos algorithmes sur des données concrètes, peu structurées, et sur un cube contenant peu de valeurs et seulement trois dimensions.

Pour chaque base, nous utilisons comme cube de départ de nos algorithmes heuristiques un cube dont les dimensions sont réorganisées aléatoirement (c'est le cas également pour l'AG). Nous mesurons l'homogénéité de ce cube initial pour avoir une valeur de référence. Nous utilisons également cette valeur de référence pour calculer un gain d'homogénéité Ben Messaoud (2006) : $\frac{H_g(final)-H_g(init)}{H_g init}$, où $H_g(final)$ représente l'homogénéité du meilleur résultat retourné par l'algorithme testé.

1. http ://kdd.ics.uci.edu/databases/census-income/census-income.html

5.2 Résultats sur la base Jouet

Nous indiquons dans le tableau 1 les résultats quantitatifs obtenus. Tout d'abord on constate que tous les algorithmes améliorent le critère H_g qui vaut 8.8% pour la matrice mélangée aléatoirement. Parmi les heuristiques, H_{Extrem} est celle qui donne les meilleurs résultats, avec un score quasiment identique à BEA. Elle trouve une matrice meilleure que la matrice cible ($H_g = 24\%$), ce qui vient du fait que dans cette matrice toutes les valeurs sont égales et donc les rapprocher augmente l'homogénéité globale. Les temps d'optimisation du cube sont très courts. En ce qui concerne l'AG, on constate qu'à qualité similaire, les temps mis pour obtenir le résultat sont beaucoup plus longs. Il apparaît donc que ce type de méthodes ne sont pas peut-être pas réellement adaptées pour le problème (pour une solution on-line), à moins d'accélérer l'exécution avec une nouvelle version de l'AG (en le parallélisant ou en l'hybridant avec une heuristique).

Nous donnons dans la figure 2 des exemples de visualisations des différentes matrices initiales et finales afin de vérifier si l'amélioration du critère H_g correspond bien à une amélioration de la lisibilité de la matrice. Les matrices trouvées par H_{Extrem} et l'AG nous semblent être effectivement celles qui font le mieux apparaître les blocs. La matrice mélangée, qui est donc la plus difficile à interpréter, est celle qui obtient le score H_g le plus faible.

5.3 Résultats sur Census-Income

Cette base concerne un recensement sur les revenus de la population des Etats-Unis d'Amérique de 1994 et 1995. Pour cette base, le cube initial a une moyenne de 14.16% d'homogénéité. D'après le tableau 1 et les illustrations de la figure 5.3, nous pouvons constater que l'AG obtient les meilleurs résultats, mais en un temps très long. Cela suggère d'utiliser l'AG de manière hors-ligne : cela peut diminuer les capacités interactives mais au profit de visualisations encore meilleures. Ensuite nous pouvons remarquer que H_{Extrem} est de nouveau la meilleure heuristique. H_{Tri} n'obtient pas les meilleurs résultats car le tri ne se base que sur la somme des mesures et non sur la similarité entre modalités : à somme égale il ne fait pas la différence entre des modalités similaires ou non.

A titre de comparaison supplémentaire, dans Ben Messaoud (2006) une ACM est utilisée pour réorganiser cette base. L'homogénéité obtenue est de 17.67% à partir d'un cube initial de 14.22% d'homogénéité, soit un gain de 24.26%. Nous constatons que notre approche (heuristique H_{Extrem}) obtient de meilleurs résultats (20.8% d'homogénéité avec un gain de 48.89% pour un cube initial similaire) dans un temps compétitif. L'AG améliore encore ces performances mais le temps d'exécution est beaucoup plus long que pour une ACM.

5.4 Base Mammographies

Cette base de données contient des informations sur des images médicales de lésions mammaires et nous utilisons ici ses trois dimensions. Elle contient beaucoup de valeurs $NULL$ (97% des valeurs). Afin de traiter ces valeurs, nous les avons remplacées par 0. D'une part la mesure est positive dans cette base, et d'autre part ce traitement permet de définir une distance de 0 entre deux valeurs $NULL$. A l'avenir nous pensons pouvoir adapter la mesure de distance pour traiter plus spécifiquement ces valeurs. Etant donné que les valeurs non nulles sont très éparses, le cube sera donc a priori difficile à réorganiser. Il n'est donc pas surprenant de trouver

une homogénéité initiale de 0.47%. Dans le tableau 1, on peut noter tout d'abord que certaines heuristiques dégradent le critère H_g. Ensuite, H_{Extrem} obtient de bons résultats mais elle est dépassée par le simple tri des modalités, ainsi que par BEA qui obtient les meilleurs résultats. Le grand nombre de valeurs à 0 biaise probablement le calcul de la distance entre modalités de nos heuristiques, ces dernières se retrouvant être similaire entre elles non pas à cause des valeurs définies mais plutôt des valeurs $NULL$. Cela suggère donc une extension de notre approche pour gérer de manière plus efficace ces valeurs dans le cas de bases très éparses.

5.5 Interaction avec un serveur OLAP et première visualisation en réalité virtuelle

Nous avons interfacé les différents algorithmes de réorganisation dans un serveur OLAP (de type Mondrian [2]). L'utilisateur choisit les dimensions qui l'intéressent, le niveau d'agrégation de ces dimensions et enfin l'algorithme de réorganisation (voir figure 4(a)). Le système retourne alors la visualisation calculée sous la forme d'un cube en 3D (voir figure 4(b)). On y retrouve les 3 dimensions de la base ainsi que les labels des modalités. La valeur de la mesure est représentée par un cube dont la taille dépend de la mesure (comme dans le système DBMiner). Cette représentation 3D a été intégrée dans notre plateforme de réalité virtuelle VRMiner (Azzag et al. (2005)). L'affichage a lieu sur un grand écran stéréoscopique polarisé permettant de percevoir réellement la troisième dimension (un écran cathodique standard peut aussi être utilisé ce qui représente un investissement beaucoup plus léger). L'utilisation de la stéréoscopie est importante pour lever des ambiguïtés habituelles en visualisation 3D (notamment le fait que la perspective modifie la taille des cubes : avec la stéréoscopie, lorsqu'un cube est petit, l'expert sait si cela est du à l'éloignement ou à une valeur de mesure faible). Les interactions sont en cours d'étude dans cette visualisation : nous avons implémenté une simple sélection des données pour en connaître les détails mais encore aucun moyen de manipulation graphique du cube. Enfin, l'utilisateur peut se déplacer à l'aide d'un SpacePilot (souris à 6 degrés de liberté). Cette première ébauche de représentation en environnement virtuel a pour objectif de prolonger les travaux présentés dans la section 1. Nous souhaitons ainsi à moyen terme combiner la réorganisation des dimensions avec une exploration interactive du cube en environnement virtuel, ceci dans le but de fournir une interface la plus utile et intuitive possible pour l'utilisateur.

6 Conclusion

A travers cet article nous avons proposé deux types d'algorithmes de réorganisation de cubes de données. Les résultats des algorithmes heuristiques ou de l'AG sont prometteurs et donnent des visualisations intéressantes. Les algorithmes heuristiques sont cependant plus compétitifs que l'AG dans le cadre d'une analyse en ligne car ils donnent un ratio qualité/temps de calcul pour le moment meilleur que l'AG. Nous sommes cependant convaincus que l'AG pourrait cependant être optimisé et accéléré avec une nouvelle version parallèle de l'algorithme utilisant une hybridation avec les heuristiques. On peut considérer également que si l'utilisateur est prêt à attendre plus longtemps (aspects hors ligne : les traitements sont faits lors de la

2. http ://mondrian.pentaho.org/

génération de l'hypercube) alors la lisibilité des visualisations pourraient être encore améliorée grâce à l'AG. Enfin, les algorithmes et visualisations résultants ont été intégrés et testés dans un serveur OLAP. Une première représentation en réalité virtuelle a été proposée.

Parmi les perspectives, nous souhaitons pouvoir étendre notre méthode aux dimensions organisées hiérarchiquement. Cela implique d'ajouter des contraintes dans les échanges de modalités effectués. Nous avons également mentionné le fait qu'un seul critère ne peut sans doute pas mesurer entièrement l'intérêt d'une visualisation, et probablement qu'aucun critère ne peut atteindre la perfection (la perception de l'expert lui-même). Cependant, nous pensons définir d'autres critères de ce type, et espérer ainsi produire des visualisations les plus proches possible de ce que recherche l'expert. Pour cela, il faudra réaliser une optimisation multicritères du cube, et utiliser l'optimalité de Pareto. Enfin, nous allons proposer de nouvelles visualisations permettant de représenter plus de trois dimensions et nous allons nous intéresser aussi aux interactions possibles dans la visualisation. Nous souhaitons que l'utilisateur puisse retrouver les opérateurs classiques d'OLAP. Notre objectif final est d'intégrer toutes ces méthodes dans notre outil VRMiner afin de proposer une interface complète en réalité virtuelle pour OLAP.

Remerciements

Nous tenons à remercier Riadh Ben Messaoud pour nous avoir transmis les données utilisées dans nos tests, ainsi que Julien Galine pour son aide dans le développement de notre plateforme d'interrogation et de visualisation.

Références

Ammoura, A., O. Zaıane, et R. Goebel (2001). Towards a Novel OLAP Interface for Distributed Data Warehouses. pp. 174–185.

Azzag, H., F. Picarougne, C. Guinot, et G. Venturini (2005). Vrminer : A tool for multimedia database mining with virtual reality. *Processing and Managing Complex Data for Decision Support.*, 318–339.

Ben Messaoud, R. (2006). *Couplage de l'analyse en ligne et de la fouille de données pour l'exploration, la structuration et l'explication des données multidimensionnelles*. Ph. D. thesis, Université Lumière Lyon 2.

Benzécri, J. P. (1973). *L'analyse des correspondances*. Paris : Dunold.

Bertin, J. (1977). La graphique et le traitement graphique de l'information. *Nouvelle Bibliothèque Scientifique.*.

Bulusu, P. (2003). *DIVA - Data Warehouse Interface for Visual Analysis*. Ph. D. thesis, UNIVERSITY OF FLORIDA.

Caraux, G. (1984). Réorganisation et représentation visuelle d'une matrice de données numériques : un algorithme itératif. *Revue de Statistique Appliquée 32*(4), 5–23.

Choong, Y. W., D. Laurent, et P. Marcel (2003). Computing Appropriate Representations for Multidimensional Data. *Data & knowledge Engineering Journal 45*(2), 181–203.

Climer, S. et W. Zhang (2006). Rearrangement Clustering : Pitfalls, Remedies, and Applications. *The Journal of Machine Learning Research 7*, 919–943.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

Han, J., Y. Fu, W. Wang, J. Chiang, W. Gong, K. Koperski, D. Li, Y. Lu, A. Rajan, N. Stefanovic, B. Xia, et O. R. Zaiane (1996). DBMiner : A System for Mining Knowledge in Large Relational Databases. pp. 250–255.

Hanan, M. et J. Kurtzberg (1972). A Review of the Placement and Quadratic Assignment Problems. *SIAM Review 14*(2), 324–342.

Holland, J. (1975). *Adaptation in natural and artificial systems*. MIT Press Cambridge, MA, USA.

Kulezynski, S. (1927). Die pflanzenassoziation der pieninen.

Kusiak, A. (1987). The generalized group technology concept. *International Journal of Production Research 25*(4), 561–569.

Lenstra, J. (1974). Clustering a data array and the traveling-salesman problem. *Operations Research 22*(2), 413–414.

McCormick, W., P. Schweitzer, et T. White (1972). Problem decomposition and data reorganization by a clustering technique. *Operations Research 20*(5), 993–1009.

Périn, P. et R. Legoux (1980). *La datation des tombes mérovingiennes : historique, méthodes, applications*. Droz.

Robardet, C. (2002). *Contribution à la classification non supervisée : proposition d'une méthode de bi-partitionnement*. PhD thesis, University Lyon 1 Claude Bernard, F-69622 Villeurbanne cedex, 2002.

Sismanis, Y., A. Deligiannakis, N. Roussopoulos, et Y. Kotidis (2002). Dwarf : Shrinking the PetaCube. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'2002)*, Madison, Wisconsin, USA, pp. 464–475. ACM Press.

Summary

In this paper we study how to reorganize data in an OLAP analysis in order to improve the visualization presented to experts. After the description of existing approaches for matrix and OLAP cube reorganization, we present two methods. The first one is a heuristic method that reorganizes the dimensions by maximizing the similarity between adjacent data. The second method is a genetic algorithm which evolves the cube by exchanging values in order to maximize the visualization evaluation criterion. We compare these two methods with other approaches on real world databases in order to optimize matrices or cubes visualizations (i.e. 2D or 3D) and we present the obtained visualizations. We detail the integration of these methods in an OLAP server and a first visualization in a virtual reality environment.

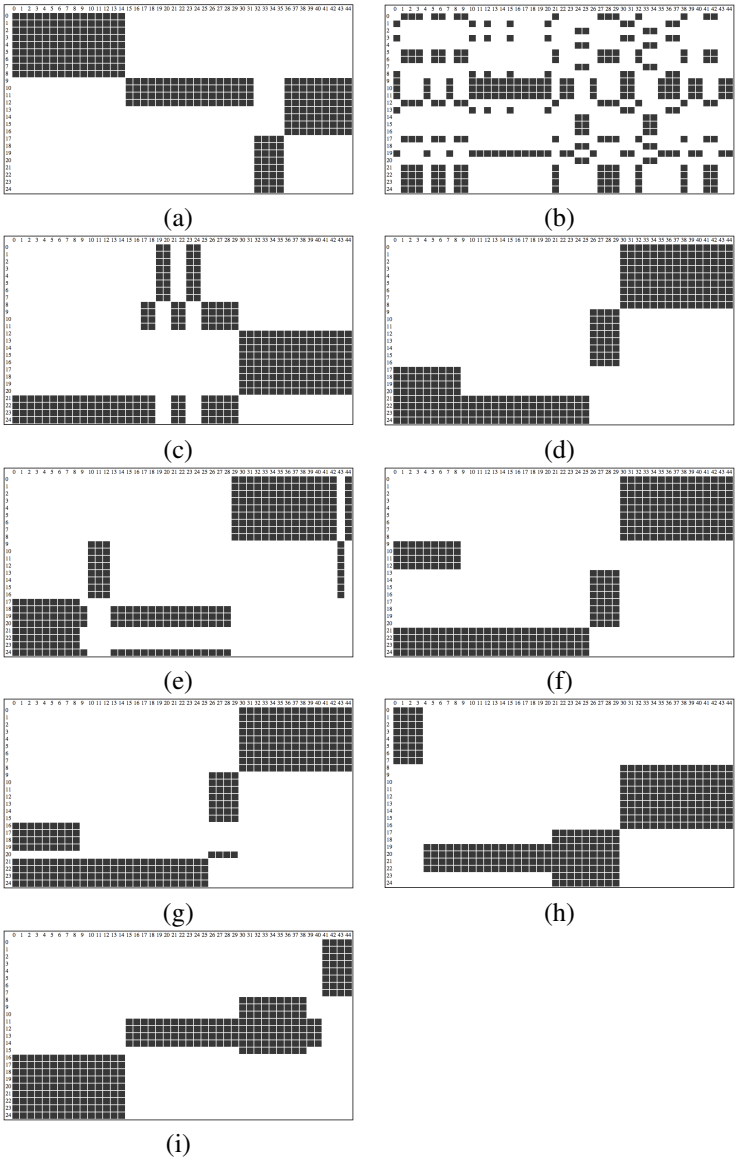
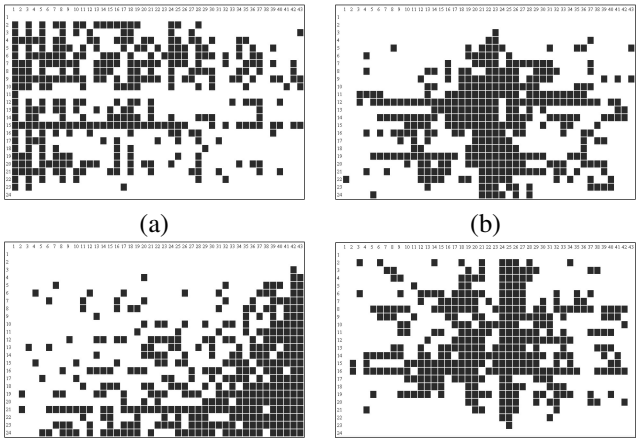


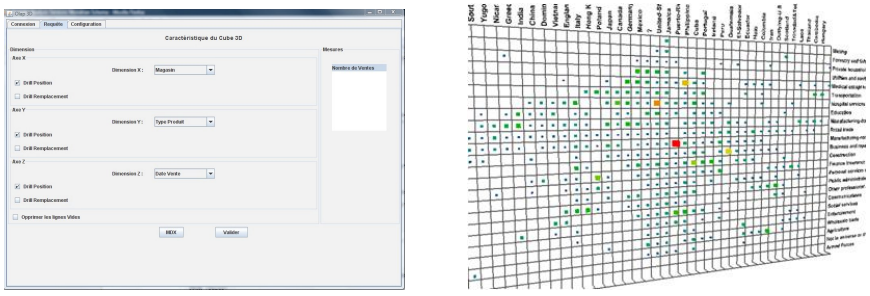
FIG. 2 – *Matrice de départ en (a), matrice mélangée en (b) et matrices obtenues par H_{tri} en (c), H_{Extrem} en (d), $H_{InsertMax}$ en (e), $H_{InsertMin}$ en (f), $H_{InsertMoy}$ en (g), heuristique BEA en (h) et AG en (i).*



(a) (b)

(c) (d)

FIG. 3 – *Résultats obtenus sur la base Census-Income : base initiale en (a), puis optimisée par BEA en (b), par l'heuristique H_{Tri} en (c) et par H_{Extrem} en (d).*



(a) (b)

FIG. 4 – *Interface de notre outil avec un serveur OLAP (a) et première ébauche de visualisation en réalité virtuelle (b).*