RAIRO Operations Research
RAIRO Oper. Res. **36** (2002) 101-108
DOI: 10.1051/ro:2002001

AGRÉGATION DES SIMILARITÉS : UNE SOLUTION OUBLIÉE

Michel Petitjean[1]

Communiqué par I.C. Lerman

Abstract. The signed similarities aggregation problem is solved with a boolean method derived from the Faure and Malgrange algorithm. The method is adequate either for integer similarities or real similarites, and multiple solutions can be enumerated. It needs a space amount equal to three times the input data size.

Résumé. Le problème de l'agrégation des similarités signées est résolu à l'aide d'une version dérivant de l'algorithme booléen de Faure et Malgrange. La méthode s'applique à des similarités entières ou réelles, et permet l'énumération des solutions multiples. Elle nécessite une quantité de mémoire égale à trois fois la taille des données d'entrée.

Mots clés : Agrégation des similarités, partition optimale, programmation linéaire en variables booléennes.

1. Introduction

Soient n individus, pour lesquels on dispose d'un tableau carré symétrique S de taille $(n,\ n)$ contenant des similarités signées, que l'on assimilera à des coûts, pouvant être indifféremment des entiers ou des réels. Le problème d'agrégation des similarités consiste à chercher une partition des n individus, qui soit optimale

Reçu le 25 janvier 2001.

[1] ITODYS, ESA 7086 du CNRS, 1 rue Guy de la Brosse, 75005 Paris, France ;
e-mail : petitjean@itodys.jussieu.fr

© EDP Sciences 2002

au sens suivant :

$$Z = \mathrm{Max}\left(\sum_i \sum_j P_{ij} \cdot S_{ij}\right) \tag{1}$$

Z est une valeur à maximiser en fonction de la matrice de partition inconnue P, de taille (n, n). On cherche donc n^2 éléments booléens P_{ij} inconnus, tels que $P_{ij} = 1$ lorque i et j sont dans la même classe, et $P_{ij} = 0$ sinon. En théorie, Z est maximal chaque fois que l'on attribue la valeur $P_{ij} = 1$ pour une similarité S_{ij} positive, et $P_{ij} = 0$ pour une similarité négative. Malheureusement ce maximum théorique n'est pas en général atteint, car les éléments de P ne sont pas indépendants, en raison des contraintes de transitivité : si i est dans la même classe que j et j dans la même classe que k, alors i et k doivent être dans la même classe. Autrement dit, $P_{ij} = 1$ et $P_{jk} = 1$ implique $P_{ik} = 1$. Notons également que S et P sont symétriques, et que les éléments diagonaux de P, qui valent forcément 1, sont sans intérêt.

Plusieurs problèmes d'analyse de données peuvent conduire à résoudre une équation de ce type [5,6]. Le plus commun d'entre eux consiste à chercher une partition optimale de n individus décrits par p variables à catégories (voir annexe). Quelle que soit la manière dont on obtient S, il est impératif de noter que le nombre de classes de la partition inconnue P est calculé, sans paramètre arbitraire, ce qui rend la méthode extrêmement attractive. Notre propos n'est cependant pas de discuter cette méthode, mais de résoudre l'équation (1). En effet, le principal auteur de la méthode [5] a utilisé une méthode du simplexe en transcrivant les contraintes de transitivité sous formes d'inéquations, comme par exemple, pour 3 individus distincts i, j, k : P_{ij}+P_{jk}-$P_{ik} \geq 1$. Il y a autant d'inconnues que de couples d'individus, et 3 fois plus de contraintes que de triplets d'individus, soit $n(n-1)/2$ inconnues et $n(n-1)(n-2)/2$ contraintes, ce qui nécessite de réserver un espace de travail proportionnel à n^5 pour manipuler le tableau du simplexe. Le couplage d'heuristiques avec une méthode dérivée du simplexe a également été utilisé [3]. Nous proposons ici une solution qui se rattache formellement à l'algorithme booléen de Faure et Malgrange [11], mais qui n'utilise plus la formulation linéaire des contraintes, et qui a pour originalité de permettre l'énumération des solutions multiples en utilisant un espace de travail seulement égal à $3n^2$ mots. Bien que la programmation dynamique ait été préconisée pour un problème voisin [1], c'est à notre connaissance, depuis la création de la méthode il y a une vingtaine d'années, la première fois qu'une solution de ce type est mise en œuvre.

2. Méthode

En examinant le panorama des méthodes de programmation linéaire en variables entières [2, 9, 11, 13, 14] on s'aperçoit que la plupart nécessitent la connaissance initiale de l'optimum continu. Dans ce cas, il faut modifier dynamiquement un tableau de taille $O(n^5)$. Ces méthodes sont en pratique limitées au traitement

de quelques dizaines d'individus, et il en est de même pour toutes les méthodes nécessitant de modifier dynamiquement le tableau du simplexe, bien que ce tableau soit assez creux au départ. La méthode de Balas [11] et la méthode de Faure et Malgrange booléenne [11] (FMB) échappent à ce besoin de modification dynamique de $O(n^5)$ éléments. Ces deux méthodes arborescentes sont conçues pour travailler en variables booléennes. En notant A la matrice des contraintes et b le second membre, les 2 méthodes nécessitent de stocker A et de modifier dynamiquement b. Par souci de clarté, nous remplacerons souvent les couples d'indices ij par un indice unique noté en majuscule : exemple, P_{ij} pourra être noté P_I.

Dans la méthode FMB, les coefficients de A peuvent être supposés non négatifs en introduisant les variables auxilliaires $\bar{P}_I = 1 - P_I$. Il en résulte que, pour les contraintes de partitionnement, A ne contient plus que des 1 et des 0, et de plus, la position des 1 dans A ne dépend que de l'ordre dans lequel les contraintes sont référencées. Donc A ne dépend plus des coûts mais seulement de n, et donc les éléments A_{IJ} sont calculables uniquement avec I et J : il est inutile de stocker A. De même, pour les contraintes de partitionnement, le vecteur b ne dépendra que des coûts : il n'est pas nécessaire de stocker b pour connaître ses éléments.

On réordonne les variables par coûts décroissants et une partition initiale est choisie. On attribue progressivement la valeur 1 ou 0 à chacune des variables, de façon à dégrader le moins possible la fonction économique. À chaque attribution d'une valeur (empilement), on examine les contraintes ; si la valeur est refusée, on attribue l'autre valeur ; si l'autre valeur est refusée, on remet en question le dernier choix effectué (dépilement). Lorsque toutes les variables sont fixées, on a un optimum local. On remet alors en question le choix de l'ultime variable, et on poursuit l'algorithme. Celui-ci s'arrête lorsque la première variable est dépilée.

L'examen des contraintes liées à l'attribution d'une valeur v à la variable P_{ij} se fait ainsi : v est refusée dès qu'il existe un indice k tel que $v + P_{ik} + P_{jk} = 2$, k pouvant varier de 1 à n. Chaque examen des contraintes représente donc un effort de calcul qui croît au plus comme la racine carrée du nombre de variables. À l'arrêt de l'algorithme, le nombre d'empilements est égal au nombre de dépilements, et le nombre de tests effectués est égal au double du nombre d'empilements.

De nombreux empilements et dépilements sont évités grâce au test suivant. À chaque attribution d'une valeur à une variable, la fonction économique possède un majorant calculé en supposant les variables libres de coût positif égales à 1 et celles de coût négatif égales à 0. Si ce majorant est inférieur au coût de la meilleure partition sauvegardée, on n'effectue pas l'empilement mais on remet en question le dernier choix effectué. L'inégalité peut être considérée au sens strict ou non, suivant que l'on souhaite trouver le plus vite possible une solution optimale, ou bien énumérer toutes les solutions.

Le choix d'une partition initiale de coût élevé permettra aussi de réduire les empilements et dépilements (voir plus loin).

En supposant que le tableau des coûts signés est lu en entrée, l'espace utilisé pour implémenter de l'algorithme comprend 6 ensembles de $n(n-1)/2$ éléments, chacun étant stocké dans la triangulaire inférieure ou supérieure d'une matrice de n fois n éléments : les coûts, les adresses des coûts renumérotés logiquement,

leurs adresses réciproques, les valeurs des variables, la sauvegarde de la partition optimale, et le chaînage des adresses de chaque dernière variable fixée. Deux diagonales sont utilisées comme buffer pour calculer le nombre et les effectifs des classes : l'une pour la partition sauvegardée et l'autre pour la partition courante.

3. Initialisation

N'importe quelle partition est une solution de départ, notamment les deux partitions triviales : $P_{ij} = 1$ pour tout couple i, j (1 classe unique de n individus), et $P_{ij} = 1$ uniquement lorsque $i = j$ (n classes de 1 individu chacune). Ces deux partitions n'ayant aucune raison d'avoir un coût élevé, nous proposons une heuristique fournissant la partition initiale par classification ascendante hiérarchique (CAH).

On définit la similarité entre deux groupes (classes) comme étant la somme des similarités individuelles de tous les couples d'éléments possibles. On initialise la CAH avec la partition triviale à n classes, et on regroupe itérativement les 2 classes ayant la plus grande similarité. Lorsque deux classes $k1$ et $k2$ sont réunies, la similarité globale entre une classe donnée $k3$ et le nouveau groupe $(k1, k2)$ est alors la somme des similarités de la classe $k3$ avec les classes séparées $k1$ et $k2$. Les itérations continuent tant qu'il existe deux classes ayant une similarité positive (classes plus "semblables" que "dissemblables"). Remarquons que le test d'arrêt conduit à n classes lorsque toutes les similarités individuelles sont négatives (il n'y a pas deux individus qui se ressemblent), mais conduit à une classe unique lorsqu'il n'y a que des similarités positives. Dans ces deux cas, la CAH donne directement la partition optimale.

L'effort de calcul pour obtenir la partition initiale est majoré par $O(n^3)$. L'une des deux triangulaires de la matrice des coûts (similarités) étant lue en entrée et l'autre triangulaire servant de buffer, la CAH écrit une des deux triangulaires de la matrice de partition, et se sert de la diagonale pour numéroter les classes.

4. Résultats

Les exemples repris dans le tableau 1 sont issus de problèmes de recherche de partition centrale de n individus décrits par plusieurs variables qualitatives (voir annexe). Ils sont tous extraits de la thèse de Marcotorchino [5].

Les $n(n-1)/2$ coûts individuels ont été triés de deux manières différentes avant d'effectuer les empilements : par valeurs algébriquement décroissantes (option 1), et par valeurs absolues décroissantes (option 2).

Le nombre $p(n)$ de partitions de n individus est calculable par sommation, à partir des nombres $p(n, k)$ de partitions de n individus en k classes, qui sont calculables par récurrence [12]. $p(n, k) = p(n-1, k-1) + k \cdot p(n-1, k)$.

Afin de ne pas alourdir la présentation, seul le premier jeu de données est complètement explicité ici : $S_{52} = S_{63} = 3$, $S_{21} = S_{43} = 1$, $S_{51} = S_{64} = -1$, $S_{61} = S_{62} = S_{65} = -3$, les autres coûts de la triangulaire inférieure valent tous

TABLEAU 1. Quelques exemples d'énumérations de partitions optimales. Les coûts concernent $n(n-1)/2$ variables, et sont en fait selon l'équation (1) des demi-coûts hors-diagonale. Le haut du tableau présente les résultats obtenus avec l'option 1, le bas du tableau avec l'option 2. $p(n)$: nombre total de partitions possibles des n individus. REF : page de référence du tableau de données dans [5]. $Z_{\max}$: borne supérieure du coût des partitions (somme des coûts S_{ij} positifs). Z_{ini} : coût de la partition initiale obtenue par CAH. Z_{opt} : coût de la partition optimale. NE1 : nombre d'empilements pour obtenir une seule partition optimale. NE : nombre d'empilements pour obtenir toutes les partitions optimales. Ncl : nombre de classes des partitions optimales, dans l'ordre de sortie.

n	$p(n)$	REF	$Z_{\max}$	Z_{ini}	Z_{opt}	NE1	NE	Ncl
6	203	p.1	8	6	6	10	49	2,3,3,4
6	203	A1, p.5-6	39	19	25	212	233	2
6	203	A1, p.7-8	43	21	24	438	492	2
8	4140	T1, p.50-54	14	13	13	16	53	3
10	115975	T2, p.15	116	92	92	673	848	5
11	678570	T3, p.11-13	63	50	50	384	525	5
30	$8{,}5\times 10^{23}$	T1, p.54-59	1400	1296	1304	994747	1123308	4,4
54	$1{,}9\times 10^{52}$	T1, p.60-61	918	791	798	?	?	6,6
6	203	p.1	8	6	6	19	22	4,3,3,2
6	203	A1, p.5-6	39	19	25	89	124	2
6	203	A1, p.7-8	43	21	24	57	65	2
8	4140	T1, p.50-54	14	13	13	27	34	3
10	115975	T2, p.15	116	92	92	137	158	5
11	678570	T3, p.11-13	63	50	50	502	580	5
30	$8{,}5\times 10^{23}$	T1, p.54-59	1400	1296	1304	7509723	8292986	4,4
54	$1{,}9\times 10^{52}$	T1, p.60-61	918	791	798	?	?	6,6

-5. Les coûts diagonaux sont égaux à 5, mais n'interviennent pas dans les calculs. Les 4 partitions optimales sont :

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

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

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

$\{1\}$, $\{2,5\}$, $\{3,6\}$, $\{4\}$· Alors que le jeu de données à $n = 30$ individus a été traité en 13 secondes sur une station VMS DEC alpha 2100 (recherche de toutes les solutions avec l'option 1), le jeu de données à 54 individus a donné deux partitions de coût optimal (selon Ref. [5]) au bout d'une demi-heure sur cette même station (près de 45 millions d'empilements), mais l'algorithme ne s'était pas terminé au bout de 12 h. En recherchant une seule solution plutôt que des les énumérer toutes, il a fallu aussi une demi-heure de calcul pour obtenir la première partition optimale (près de 43 millions d'empilements), et l'algorithme ne s'est pas non plus

terminé en 12 h ; l'option 2 aura nécessité respectivement environ 82 millions et 78 millions d'empilements pour arriver aux mêmes résultats.

5. Discussion et conclusion

Le nombre de partitions de n individus en k classes est plus grand que le coefficient du binôme lorsque $1 < k < n$. Le nombre total de partitions de n individus dépasse donc largement 2^n. Au pire, toutes les partitions peuvent être optimales (coûts nuls), donc l'énumération de toutes les solutions est un problème de nature exponentielle.

La méthode pourrait être améliorée en adoptant une meilleure heuristique d'initialisation, mais surtout en introduisant des tests plus efficaces pour éliminer les solutions non optimales lors des empilements. L'option du tri initial des coûts par valeurs algébriquement décroissantes semble meilleure pour les grandes valeurs de n.

La méthode peut être étendue au cas où l'équation (1) concerne une matrice S non symétrique. Sachant que les éléments diagonaux de P valent forcément 1, et qu'il n'y a que $n(n-1)/2$ inconnues car P est toujours symétrique, on se ramène au cas symétrique en travaillant avec la matrice $(S + S')/2$, S' étant la transposée de S. Lorsque tous les éléments de S sont non négatifs, la partition triviale à 1 classe est solution, et lorsque tous les éléments de S sont non positifs, la partition triviale à n classes est solution.

Sauf cas particuliers, la méthode proposée est inefficace au-delà d'une cinquantaine d'individus. Pour un grand nombre d'individus, les méthodes publiées précédemment [3, 5] restent donc préférables. Notre méthode présente néanmoins quelques avantages : on peut d'une part obtenir toutes les solutions optimales et non une seule, et d'autre part, grâce au faible encombrement mémoire et à la non nécessité de bibliothèque mathématique, n'importe quel utilisateur peut facilement éxécuter le programme sur une machine bas de gamme. Nous mettons le programme à disposition publique, binaires et sources (fortran 77). Une application a déjà été publiée [15], et il est envisagé d'autres applications dans le domaine de la classification de graphes [10].

Annexe : Recherche de partition centrale

Soient n individus décrits par p variables qualitatives. Chacune de ces p variables, prise isolément, partitionne les n individus, mais les p partitions n'ont aucune raison d'être cohérentes entre elles. La similarité S_{ij} entre les individus i et j est égale au nombre de fois où i et j sont classés ensemble (nombre d'accords) moins le nombre de fois où ils ne sont pas classés ensemble (nombre de désaccords). Maximiser Z dans l'équation (1) consiste à maximiser la différence : nombre d'accords moins nombre de désaccords. Pour chaque couple (i, j),

le nombre d'accords plus le nombre de désaccords étant constant (égal à p), maximiser Z revient donc à chercher la partition qui maximise le nombre d'accords : c'est le principe de majorité de Condorcet [5–8].

De plus, en notant X le tableau disjonctif à n lignes associé aux p variables, le tableau $C = X \cdot X'$ est la somme des p matrices de partition associées aux variables, et est tel que l'élément C_{ij}, qui est le nombre d'accords, vérifie : $S_{ij} = 2C_{ij} - p$. Minimisons la norme de Schur de la différence entre la partition inconnue P et la partition moyenne C/p:

$$p \cdot \mathrm{Min}\|P - C/p\|^2 = p \cdot \mathrm{Min} \sum_i \sum_j (P_{ij} - C_{ij}/p)^2 = \mathrm{Min} \sum_i \sum_j P_{ij}(p - 2C_{ij}).$$

Au signe près, on retrouve l'équation (1). La partition inconnue P est donc la plus proche de la partition moyenne [12], et constitue une partition centrale [4].

Finalement, bien qu'apparamment non mentionné dans la littérature, il est possible d'attribuer un poids w_k à la variable k en comptant les accords qui lui sont associés pour w_k au lieu de $+1$ et ses désaccords pour w_k au lieu de -1, ce qui revient à multiplier par w_k la matrice de partition associée à chaque variable k. La somme des poids peut être quelconque, et les éléments de S peuvent être des réels aussi bien que des entiers. Notons que lorsqu'une même partition est associée à plusieurs variables qualitatives, on peut les remplacer par une variable unique dont le poids est égal à la somme des poids individuels des variables.

Remerciements. Je remercie Gilbert Saporta pour m'avoir présenté la méthode d'agrégation des similarités de François Marcotorchino, et pour m'avoir incité à y apporter une nouvelle solution. Je remercie également le rapporteur de cet article pour m'avoir fourni diverses informations bibliographiques sur l'état de l'art. Enfin, je remercie Israël Lerman, membre de l'équipe éditoriale de ce journal, pour ses suggestions d'amélioration du contenu de l'article.

Références

[1] J.S. deCani, A Branch and Bound Algorithm for Maximum Likelihood Paired Comparison Ranking. *Biometrika* **59** (1972) 131-135.

[2] F. Faure, *La programmation linéaire appliquée*, Collection "Que sais-je ?", No. 1776, Chap. 4. Presses Universitaires de France, Paris (1979).

[3] M. Grötschel et Y. Wakabayashi, A Cutting Plane Algorithm for a Clustering Problem. *Math. Prog. Ser. B* **45** (1989) 59-96.

[4] I.C. Lerman, *Classification et analyse ordinale des données*, Chap. 1.3. Dunod Ed., Bordas, Paris (1981).

[5] F. Marcotorchino, *Agrégation des similarités en classification automatique*, Ph.D. Thesis. Université Paris VI, France (1981).

[6] J.-F. Marcotorchino et P. Michaud, *Optimisation an analyse ordinale des données*, Chap. X, Collection : Statistiques et décisions économiques. Masson, Paris (1979).

[7] P. Michaud, *Agrégation à la majorité II : analyse du résultat d'un vote.* Centre Scientifique IBM France, Étude F.052, Paris (1985).

[8] P. Michaud, *Hommage à Condorcet (version intégrale pour le bicentenaire de l'essai de Condorcet).* Centre Scientifique IBM France, Étude F.094, Paris (1985).

[9] M. Minoux, *Programmation mathématique. Théorie et algorithmes*, Vol. 2, Chap. 7, Collection technique et scientifique des Télécommunications. CNET-ENST Eds., Bordas, Paris (1983).
[10] M. Petitjean, Applications of the Radius-Diameter Diagram to the Classification of Topological and Geometrical Shapes of Chemical Compounds. *J. Chem. Inf. Comput. Sci.* **32** (1992) 331-337.
[11] Roseaux, *Exercices et problèmes résolus de recherche opérationnelle*, Tome 3, Chap. III. Masson, Paris (1985).
[12] G. Saporta, *Probabilités, analyse des données et statistique*, Chap. 12, Sections 12.1 et 12.2. Technip, Paris (1990).
[13] A. Schrijver, *Theory of Linear and Integer Programming*, Part IV. John Wiley and Sons, New-York (1986).
[14] G. Sierksma, *Linear and Integer Prpgramming. Theory and Practice*. Marcel Dekker Inc., New-York, *Monogr. and Textbooks in Pure Appl. Math.* **198** (1996).
[15] G. Vernin et M. Petitjean, Application de la méthode de recherche de partition centrale sur variables pondérées à la classification des vins. Étude préliminaire. *Rev. Fr. Oenol. (Cahier Scientifique)* **31** (1991) 7-15.

to access this journal online:
www.edpsciences.org