

Comparaison de deux clusterings par couplage entre clusters de clusters

Frédéric Cazals, Dorian Mazauric, Romain Tetley, Rémi Watrigant

▶ **To cite this version:**

Frédéric Cazals, Dorian Mazauric, Romain Tetley, Rémi Watrigant. Comparaison de deux clusterings par couplage entre clusters de clusters. ALGOTEL 2018 - 20èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications, May 2018, Roscoff, France. hal-01774440

HAL Id: hal-01774440
https://inria.hal.science/hal-01774440

Submitted on 23 Apr 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Comparaison de deux clusterings par couplage entre clusters de clusters

Frédéric Cazals[1], Dorian Mazauric[1], Romain Tetley[1] et Rémi Watrigant[2]

[1] *Université Côte d'Azur & Inria Sophia Antipolis - Méditerranée*
[2] *Université Claude Bernard Lyon 1*

Le clustering est une tâche essentielle en analyse de données. La variété des méthodes disponibles rend celle-ci ardue. Diverses stratégies ont été proposées pour analyser la stabilité d'un clustering en fonction des paramètres de l'algorithme l'ayant généré, ou pour comparer des clusterings produits par des algorithmes différents. Dans cet article, nous proposons une nouvelle classe de méthodes formant des groupes de clusters (*meta-clusters*) dans chaque clustering, et établissant une correspondance entre ceux-ci. Nous définissons le graphe d'intersection de deux clusterings en associant à chaque cluster un sommet et en reliant deux clusters dont l'intersection est non nulle par une arête pondérée par le poids de leur intersection. Étant donné une borne supérieure D sur le diamètre du graphe induit par les clusters des meta-clusters, nous formalisons et analysons le problème D-family-matching. Nous montrons que ce problème est NP-complet, développons des algorithmes de programmation dynamique pour certaines classes de graphes (arbres en particulier), et concevons des algorithmes efficaces basés sur des arbres couvrants pour des graphes généraux. Nous illustrons à travers des expériences le rôle de D comme un paramètre d'échelle apportant des informations sur la relation entre deux clusters. Nous montrons également les avantages de notre appariement face aux méthodes classiques de comparaison de clusters telles que la variation d'information (VI). Enfin, pour des clusterings générés par K-means, nous montrons que notre algorithme de programmation dynamique permet d'estimer le nombre effectif de clusters.

Mots-clefs : Comparaison de clusterings, décompositions de graphes, NP-complétude, programmation dynamique.

1 Motivation, problème et contributions

Le clustering, qui consiste à regrouper des points de données en ensembles disjoints d'éléments similaires, est une tâche fondamentale en analyse de données. De nombreuses classes de méthodes ont été développées (approches hiérarchiques [DH73], K-means [AV07], approches basées sur la densité [Che95, CM02]…). Toutefois, bien que des statistiques existent pour comparer globalement deux clusterings [Mei07], la recherche de correspondances entre les deux ensembles de clusters fait défaut. Nous pallions ce manque en présentant un modèle théorique permettant d'établir des groupements de clusters (ou meta-clusters). Notre approche est basée sur un problème d'optimisation combinatoire sur les graphes : le D-family-matching. Nous montrons que ce problème est NP-difficile mais proposons des algorithmes exacts pour certaines classes d'instances et un algorithme générique dans le cas général. Enfin, nous illustrons l'utilité d'un tel modèle en l'appliquant au grainage de K-means. Tous les détails et toutes les preuves se trouvent dans la version longue de notre travail [CMTW17].

2 Formalisation du problème et complexité

Soit $t \geq 1$ un entier. Considérons un ensemble d'éléments $Z = \{z_1, \ldots, z_t\}$ et deux clusterings F et F' de Z. Formellement, $F = \{F_1, \ldots, F_r\}$, $r \geq 1$, avec $F_i \subseteq Z$, $F_i \neq \emptyset$ et $F_i \cap F_j = \emptyset$ pour tout $i, j \in \{1, \ldots, r\}$, $i \neq j$. De manière analogue, $F' = \{F'_1, \ldots, F'_{r'}\}$, $r' \geq 1$, avec $F'_i \subseteq Z$, $F'_i \neq \emptyset$ et $F'_i \cap F'_j = \emptyset$ pour tout $i, j \in \{1, \ldots, r'\}$, $i \neq j$. La Figure 1 (a) décrit deux clusterings F et F' avec $t = 40$, $r = 2$ et $r' = 5$. Le graphe d'intersections arête-pondéré $G = (V, E, w)$ associé à Z, F et F', est tel que $V = \{F_1, \ldots, F_r\} \cup \{F'_1, \ldots, F'_{r'}\}$, $E = \{\{F_i, F'_j\} \mid F_i \cap F'_j \neq \emptyset, 1 \leq i \leq r, 1 \leq j \leq r'\}$ et le poids d'une arête $e = \{F_i, F'_j\} \in E$ est $w_e = |F_i \cap F'_j|$. Autrement dit, les sommets représentent les clusters et le poids d'une arête est le nombre d'éléments communs aux deux clusters correspondant aux deux sommets (si cette intersection est vide, alors il n'y a pas d'arête). Ainsi, tout

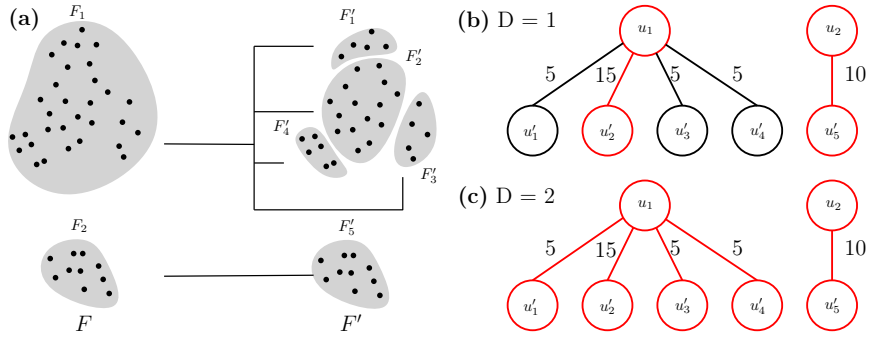


FIGURE 1: Comparaison de deux clusterings du même ensemble de données 2D composé de 40 points. (a) Clustering F composé de 2 clusters de 30 et 10 points. Clustering F' composé de 5 clusters de 5, 15, 5, 5 et 10 points. Le graphe d'intersections associé aux deux clusterings est représenté dans (b) et (c) : un sommet par cluster, une arête entre deux sommets si les clusters correspondants partagent au moins un point, le poids d'une arête est le nombre de points en commun. Notre méthode regroupe les clusters en meta-clusters et est paramétré par le diamètre D des sous-graphes connectant les clusters à l'intérieur des meta-clusters (en rouge). Les méthodes existantes sont basées sur les couplages maximaux d'un graphe et correspondent au cas $D = 1$. **(b)** Avec $D = 1$, un couplage est obtenu : F_1 avec F'_2 et F_2 avec F'_5. **(c)** Avec $D = 2$, $\{F_1\}$ est couplé avec le meta-cluster $\{F'_1, F'_2, F'_3, F'_4\}$ et $\{F_2\}$ est couplé avec $\{F'_5\}$.

graphe d'intersections est un graphe biparti. De plus, on peut montrer que tout graphe biparti est le graphe d'intersections de deux clusterings d'un ensemble Z. La Figure 1 (b,c) représente le graphe d'intersections de l'exemple décrit précédemment. Nous définissons maintenant la notion de D-family-matching.

Définition 1 (D-family-matching) *Soient $D \geq 1$ un entier positif et $G = (V, E, w)$ un graphe d'intersections de deux clusterings. Un D-family-matching pour G est une famille $\mathcal{S} = \{S_1, \ldots, S_k\}$ telle que pour tout $i, j \in \{1, \ldots, k\}$, $i \neq j$, alors $S_i \subseteq V$, $S_i \neq \emptyset$, $S_i \cap S_j = \emptyset$, et le graphe $G[S_i]$ induit par l'ensemble de sommets S_i a diamètre au plus D. Le score $\Phi(\mathcal{S})$ d'un D-family-matching $\mathcal{S}$ est $\Phi(\mathcal{S}) = \sum_{i=1}^{k} \sum_{e \in E(G[S_i])} w_e$.*

Étant donné un graphe d'intersections G, le problème D-family-matching consiste à déterminer $\Phi_D(G) = \max_{\mathcal{S} \in \mathcal{S}_D(G)} \Phi(\mathcal{S})$ avec $\mathcal{S}_D(G)$ l'ensemble de tous les D-family-matching pour G. Intuitivement, le problème revient à déterminer un D-family-matching qui minimise les inconsistences. La Figure 1 décrit une instance simple du problème D-family-matching et des solutions optimales pour différentes valeurs de D. Observons que pour $D = 1$, le problème est équivalent à celui du couplage de poids maximum dans les graphes bipartis. En effet, comme G est biparti, tout sous-graphe de diamètre au plus 1 est nécessairement une arête ou un sommet. Comme le problème du couplage maximum admet un algorithme de complexité $O(n^2 \log n + nm)$ [FT87], nous en déduisons la même complexité pour le problème 1-family-matching. Comme nous allons le voir, la complexité du problème est différente si $D > 1$.

Nous prouvons dans le Théorème 1 que le problème est NP-difficile dès que le diamètre est au moins 2.

Théorème 1 *Soit $D \geq 2$. Le problème de décision du problème D-family-matching est NP-complet pour :*

- *les graphes bipartis de degré maximum 3 (avec des poids unitaires si $D = 2$) ;*
- *les graphes bipartis de degré maximum 4 même si le poids maximum est constant.*

Concernant l'approximabilité du problème, une approche naturelle serait de choisir de manière gloutonne une collection de sous-graphes de diamètre D. Cette approche, qui donne une 2-approximation dans le cas $D = 1$ pour des graphes non bipartis de degré maximum constant, échoue malheureusement dans notre cas pour $D = 2$. Une autre stratégie serait alors de partir d'une solution (exacte ou approchée) du problème $(D-1)$-family-matching afin d'obtenir un D-family-matching. Malheureusement, il est possible de construire des instances pour lesquelles une solution optimale avec des meta-clusters de diamètre $D-1$ est arbitrairement plus mauvaise qu'une solution optimale avec des meta-clusters de diamètre D. La question de l'existence d'un algorithme d'approximation (avec un rapport constant) reste ouverte.

3 Algorithmes exacts polynomiaux pour les arbres et les cycles

Nous prouvons dans le Théorème 2 des algorithmes polynomiaux lorsque le graphe d'intersections est une union d'arbres, de chemins ou de cycles. Le nombre de sommets est noté n.

Théorème 2 *Soit $D \geq 1$. Il existe un algorithme résolvant optimalement D-family-matching en temps*

- $O(D^2\Delta^2 n)$ *si G est une union disjointe d'arbres de degré maximum* Δ*,*
- $O(Dn)$ *si G est une union disjointe de chemins,*
- $O(D^2 n)$ *si G est une union disjointe de cycles.*

Esquisse de preuve pour le cas d'un arbre. Soit T_r un arbre enraciné en $r \in V$. Pour tout $v \in V$, soit T_v le sous-arbre de T_r enraciné en v tel que $V(T_v)$ contient tous les sommets $v' \in V$ tels qu'il existe un chemin simple entre v' et r dans T_r qui contient v. Nous définissons la fonction Ψ_D comme suit. Pour tout $v \in V$ et tout $i \in [\![-1, D]\!]$, $\Psi_D(T_v, i)$ est le score d'une solution optimale $\mathcal{S}$ pour le problème D-family-matching pour T_v, telle que le sous-arbre induit par l'ensemble de sommets $S \in \mathcal{S}$, $v \in S$, a profondeur au plus i (si $i = -1$, alors pour tout $S \in \mathcal{S}$, nous avons $v \notin S$). Notons que $\Psi_D(T_v, 0)$ est le score d'une solution optimale $\mathcal{S}$ lorsque $\{v\} \in \mathcal{S}$ (v est seul dans un ensemble). Dans la suite, nous écrirons $\Psi_D(v, i)$ à la place de $\Psi_D(T_v, i)$. Tout d'abord, pour toute feuille $v \in V$ de T_r et tout $i \in [\![-1, D]\!]$, alors $\Psi_D(v, i) = 0$. Une feuille est un sommet de degré un et différent de la racine r. Considérons à présent un sommet $v \in V$ qui n'est pas une feuille. Soit $N(v) = \{v_1, \ldots, v_q\}$ l'ensemble des $q \geq 1$ voisins de v dans T_v. Supposons que $\Psi_D(v_j, i)$ a été calculé pour tout $j \in [\![1, q]\!]$ et tout $i \in [\![-1, D]\!]$. Nous en déduisons $\Psi_D(v, i)$ pour tout $i \in [\![-1, D]\!]$ selon les deux cas suivants (détails dans [CMTW17]).
Pour tout $i \in \{-1, 0\}$, alors

$$\Psi_D(v, i) = \sum_{j \in [\![1,q]\!]} \max_{i \in [\![0,D]\!]} \Psi_D(v_j, i).$$

Pour tout $i \in [\![1, D]\!]$, alors

$$\Psi_D(v, i) = \max_{j \in [\![1,q]\!]} (\Psi_D(v_j, i-1) + w_{v,v_j} + \sum_{j' \in [\![1,q]\!] \setminus \{j\}} \max(\max_{i' \in [\![1, D-i-1]\!]} \Psi_D(v_{j'}, i') + w_{v,v_{j'}}, \max_{i' \in [\![1,D]\!]} \Psi_D(v_{j'}, i'))).$$

Pour tout $v \in V$, la complexité de calculer $\Psi_D(v, i)$, pour tous les $i \in \{-1, \ldots, D\}$, est $O(qD)$ dans le premier cas et $O(q^2 D^2)$ dans le deuxième cas. Nous obtenons que la complexité en temps de l'algorithme est $O(D^2\Delta^2 n)$. Notons que $\Delta \leq n-1$ et $D \leq n-1$. Enfin, lorsque nous avons calculé $\Psi_D(r, i)$ pour tout $i \in \{-1, \ldots, D\}$, nous pouvons déduire une solution optimale $\mathcal{S}$ pour le problème D-family-matching pour T_r. Plus précisément, $\Phi(\mathcal{S}) = \Phi_D(=) \max_{i \in [\![-1,D]\!]} \Psi_D(r, i)$.

4 Algorithmes efficaces pour les graphes généraux

Dans cette section, nous développons un algorithme générique pour notre problème, qui consiste à considérer des arbres couvrants de G (de manière séquentielle) et de calculer des D-family-matching pour G en se basant sur ces arbres. Les trois ingrédients principaux sont les suivants :

- *Un générateur d'arbres couvrants* $\mathcal{R}(G, t)$. Cette fonction calcule l'arbre couvrant enraciné T^t de G qui est utilisé à l'étape $t \geq 1$ par l'Algorithme $\mathcal{A}$.
- *Un algorithme* $\mathcal{A}(G, T^t, D)$. Cet algorithme calcule un D-family-matching $\mathcal{S}^t$ pour G basé sur l'arbre couvrant $\mathcal{R}(G, t) = T^t$ de G. L'algorithme $\mathcal{A}(G, T^t, D)$ peut être celui décrit dans le Théorème 2.
- *Une propriété d'arrêt* $\Pi(\mathcal{M})$. Cette dernière dépend de l'ensemble de solutions $\mathcal{M}$ calculées précédemment. Tant que celle-ci n'est pas satisfaite, nous générons un autre arbre couvrant enraciné T^t de G (en utilisant $\mathcal{R}$) et calculons un D-family-matching $\mathcal{S}^t$ pour G basé sur T^t (en utilisant $\mathcal{A}$).

Nous avons également prouvé un algorithme de programmation dynamique sur un arbre couvrant mais qui prend également en compte les autres arêtes du graphe (qui ne sont pas dans l'arbre). Nous avons montré qu'il existe au moins un arbre couvrant tel que cet algorithme retourne une solution optimale au problème. Le prix à payer est de ne plus avoir nécessairement une complexité polynomiale. Voir [CMTW17] pour plus de détails. Une question intéressante est de savoir pour quelles classes d'instances, il existe un résultat analogue (exact ou approximation) pour l'algorithme polynomial décrit dans le Théorème 2.

5 Illustration de nos méthodes pour l'instabilité de K-means

L'algorithme générique de la Section 4 a été implémenté en C++ dans la couche *Core* de la Structural Bioinformatics Library (`http://sbl.inria.fr`), une librairie C++ combinant des algorithmes bas-niveau avec des applications en biologie structurale. Les paramètres utilisés sont les suivants : $\Pi(\mathcal{M})$ est vraie si

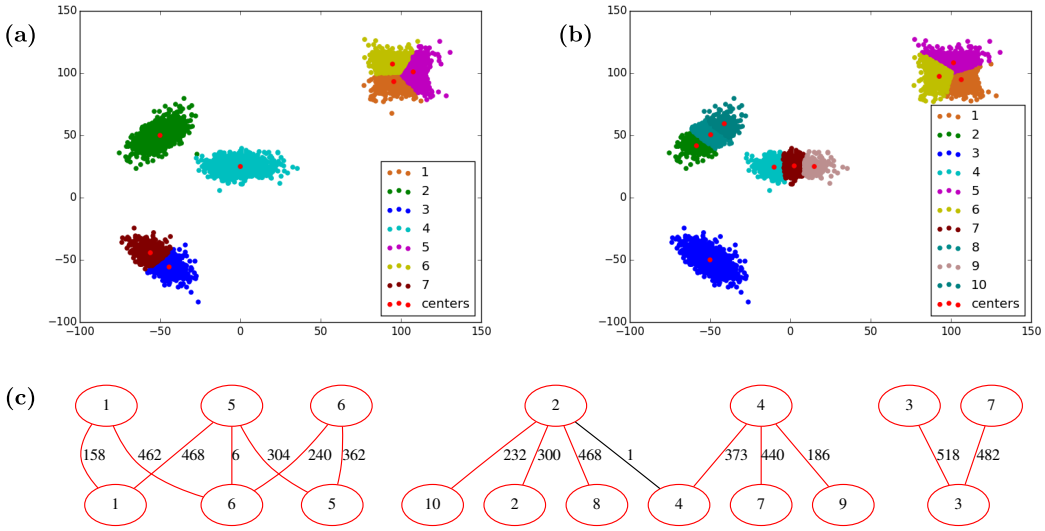


FIGURE 2: Instabilité de K-means avec smart seeding : illustration sur un ensemble de 5k points 2D calculés à partir d'un mélange de 5 gaussiennes. **(a)** K-means pour $K = 7$. **(b)** K-means pour $K = 10$. **(c)** Le résultat de notre algorithme pour $D = 3$ superposé sur le graphe d'intersections des deux clusterings. Les labels des clusters suivent les légendes des figures supérieures. Les quatre meta-clusters sont représentés en rouge.

et seulement si les scores des $p \geq 2$ meilleures solutions sont dans un intervalle de taille au plus ϕ (pour un ϕ donné), le générateur $\mathcal{R}(G,t)$ génère des arbres couvrants de manière aléatoire uniforme et l'algorithme $\mathcal{A}(G,T^t,D)$ est celui du Théorème 2 avec une étape supplémentaire consistant à ajouter séquentiellement les arêtes qui ne sont pas dans l'arbre couvrant et qui ne violent pas les contraintes.

L'ensemble des expériences conduites ne seront pas detaillées ici et peuvent être trouvées dans la version longue de ce travail [CMTW17]. Pour illustrer l'intérêt du problème D-family-matching , nous présentons une expérience réalisée sur des clusterings obtenus à partir de l'algorithme K-means en utilisant la stratégie de *smart seeding*. Nous avons tout d'abord généré un ensemble aléatoire de 5 000 points $2D$ en mélangeant 5 gaussiennes. Sur cet ensemble, nous avons généré deux clusterings avec $K = 7$ et $K = 10$, respectivement (Figure 2 (a,b)). Observons que deux des gaussiennes sont mélangées en un seul nuage de points (en haut à droite dans les figures). Lorsque le nombre de centroïdes est supérieur au nombre exact de clusters (4 dans notre situation), ces derniers sont séparés de manière arbitraire. Notre algorithme retrouve les 4 clusters originaux (Figure 2 (c)). Le score de la solution est $\Phi = 4\ 999$ car un point est attribué au mauvais cluster dans un des clusterings (Figure 2 (a)). Cet exemple démontre l'importance du paramètre D : pour $D \geq 4$, notre algorithme retourne 3 meta-clusters (avec un score $\Phi = 5\ 000$ mais au détriment de retrouver les clusters) et pour $D = 1$, notre algorithme retourne 6 meta-clusters (avec un score $\Phi = 2\ 718$). La difficulté est de déterminer le diamètre à utiliser (en pratique les algorithmes sont exécutés pour différents diamètres).

Références

[AV07] D. Arthur and S. Vassilvitskii. k-means++ : The advantages of careful seeding. In *ACM-SODA*, page 1035. Society for Industrial and Applied Mathematics, 2007.

[Che95] Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE PAMI*, 17(8) :790–799, 1995.

[CM02] D. Comaniciu and P. Meer. Mean shift : A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5) :603–619, 2002.

[CMTW17] F. Cazals, D. Mazauric, R. Tetley, and R. Watrigant. Comparing two clusterings using matchings between clusters of clusters. 2017. Inria tech report 9063.

[DH73] R.O. Duda and P.E. Hart. *Pattern classification and scene analysis*. Wiley, 1973.

[FT87] M. Fredman and R. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3) :596–615, July 1987.

[Mei07] M. Meilă. Comparing clusterings–an information based distance. *Journal of multivariate analysis*, 98(5) :873–895, 2007.