Comparing two clusterings using matchings between clusters of clusters

Frédéric Cazals, Dorian Mazauric, Romain Tetley, Rémi Watrigant

Comparing two clusterings using matchings between clusters of clusters

Frédéric Cazals and Dorian Mazauric and Romain Tetley and Rémi Watrigant

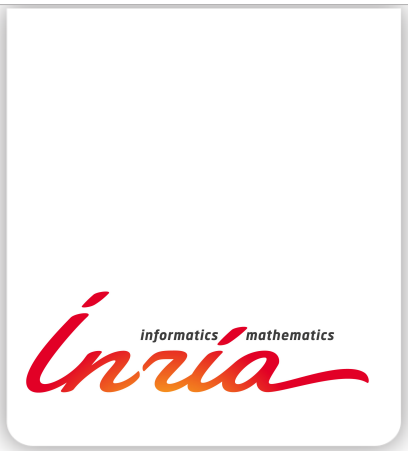
RESEARCH REPORT

N° 9063

August 2017

Project-Team Algorithms-Biology-Structure

ISSN 0249-6399 ISRN INRIA/RR--9063--FR+ENG



Comparing two clusterings using matchings between clusters of clusters

Frédéric Cazals and Dorian Mazauric and Romain Tetley and Rémi Watrigant

Project-Team Algorithms-Biology-Structure

Research Report n° 9063 — version 4 — initial version August 2017 — revised version July 2019 — 48 pages

Abstract: Clustering is a fundamental problem in data science, yet, the variety of clustering methods and their sensitivity to parameters make clustering hard. To analyze the stability of a given clustering algorithm while varying its parameters, and to compare clusters yielded by different algorithms, several comparison schemes based on matchings, information theory and various indices (Rand, Jaccard) have been developed. We go beyond these by providing a novel class of methods computing meta-clusters within each clustering– a meta-cluster is a group of clusters, together with a matching between these.

Let the intersection graph of two clusterings be the edge-weighted bipartite graph in which the nodes represent the clusters, the edges represent the non empty intersection between two clusters, and the weight of an edge is the number of common items. We introduce the so-called D-family-matching problem on intersection graphs, with D the upper-bound on the diameter of the graph induced by the clusters of any meta-cluster. First we prove NP-completeness results and unbounded approximation ratio of simple strategies. Second, we design exact polynomial time dynamic programming algorithms for some classes of graphs (in particular trees). Then, we prove spanning-tree based efficient algorithms for general graphs.

Our experiments illustrate the role of D as a scale parameter providing information on the relationship between clusters within a clustering and in-between two clusterings. They also show the advantages of our built-in mapping over classical cluster comparison measures such as the variation of information (VI).

Key-words: Clustering stability, comparison of clusterings, graph decomposition, NP-completeness, dynamic programming algorithms

RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Comparer deux clusterings en utilisant des clusters de clusters

Résumé : Le clustering est une tâche essentielle en analyse de données, mais 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 bien comparer des clusterings produits par des algorithmes différents. Nous allons au delà de celles-ci, en proposant une nouvelle classe de méthodes formant des groupes de clusters (*meta-clusters*) dans chaque clustering, et établissant une correspondance entre ceux-ci.

Plus spécifiquement, définissons le graphe intersection de deux clusterings comme le graphe biparti dont les sommets sont les clusters, chaque arête étant pondérée par le nombre de points communs à deux clusters. Nous définissons le D-family-matching problème à partir du graphe intersection, D étant une borne supérieure sur le diamètre du graphe induit par les clusters des meta-clusters. Dans un premier temps, nous établissons des résultats de difficulté et d'inaproximabilité. Dans un second temps, nous développons des algorithmes de programmation dynamique pour certaines classes de graphes (arbres en particulier). Enfin, nous concevons des algorithmes efficaces, basés sur des arbres couvrants, pour des graphes généraux.

Nos résultats expérimentaux illustrent le rôle de D comme un paramètre d'échelle fournissant de l'information sur la relation entre les clusters intra ou inter clusterings. Ils montrent aussi les avantages de notre appariement sur les outils de comparaison de clusterings classiques comme la variation d'information (VI).

Mots-clés : Stabilité du clustering, comparaison de clusterings, décompositions de graphes, NP-complétude, programmation dynamique

Contents

1 Introduction

1.1 Clusterings: generation, comparison and stability assessment

Clustering methods. Clustering, namely the task which consists in grouping data items into dissimilar groups of similar elements, is a fundamental problem in data analysis at large [46]. Existing clustering methods may be ascribed to the following categories. *Hierarchical clustering* methods typically build a dendrogram whose leaves are the individual items, the grouping aggregating similar clusters [16]. *k-means and variants* perform a grouping induced by the Voronoi cells of the cluster representatives, which are updated in an iterative fashion [2]. In *density based clustering* methods, a density estimate is typically computed from the data, with clusters associated to the catchment basins of local maxima [7]. Topological persistence may be used to select the significant maxima [6]. The notions of culminance and proeminance, which have been used since the early days of topography by geographers to define summits on mountains, can also be used to define clusters [37]. Finally, *spectral clustering* methods define clusters from the top singular vectors of the matrix representing the data (or their similarity) [43]. Each of these categories comes with its intrinsic difficulties. For example, the smart seeding strategy of k-means yields an algorithm with an approximation guarantee [2]; yet, k-means++ still suffers from instabilities when the number of centers used is larger than the *exact* number of clusters, as the clustering obtained depends on the initial distribution of centers within the clusters [43].

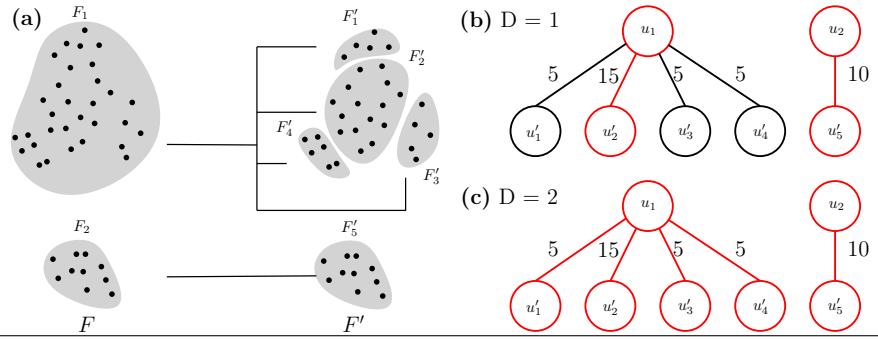
This type of difficulty together with the vast array of clustering schemes actually raises two important questions: comparing clusterings–including the problem of finding many-to-many correspondences between clusters, and estimating the number of clusters. We briefly review previous work on these.

Comparing clusterings. To describe existing cluster comparison methods, we consider two clusterings F and F' of some data set $Z = \{z_1, \ldots, z_t\}$ composed of t items. Recall that the contingency table of F and F' is the matrix in which a cell counts the number of data items common to any two clusters from F and F'. For the sake of exposure, we define a *meta-cluster* of a clustering as a set of clusters of this clustering.

In *set matching based comparisons* [29, 13], a *greedy best effort* 1-to-1 matching between clusters is sought from the contingency table. A statistic is designed by adding up the contributions of these pairs. The resulting measure is often called the *minimal matching distance (MMD)* [30]. To define MMD for the k-means algorithms, assuming that k clusters are produced, each identified by a label, denote $\Pi = \{\pi\}$ the set of all permutations of the k labels. The MMD is defined by $d_{MMD}(F, F') = \frac{1}{t} \min_{\pi \in \Pi} \sum_{i=1}^{t} \mathbf{1}_{F(x_i) \neq \pi(F'(x_i))}$, where $F(x_i)$ ($F'(x_i)$, respectively) is the cluster of F (F', respectively) containing x_i. Finding the best permutation reduces to a maximum perfect matching, and thus has polynomial complexity. Likewise, to compare clusterings with different numbers of clusters, one computes a maximum weight bipartite matching. However, MMD is inherently based on a 1-1 mapping between clusters, a stringent condition we shall get rid off–MMD shall be covered by the diameter constraint $D = 1$ in our framework (Fig. 1).

In *pair counting methods* [44, 32], each pair of items is ascribed to a category out of four (in the same cluster in F and F', in different clusters in F and F', in the same cluster of F but in different clusters in F', and vice versa). The sizes of these four classes can be used in several ways [44], including the Chi square coefficient, the Rand Index, the Fowlkes-Mallows index, the Mirkin metric, etc. However, with a focus on pairs, such methods do not provide any insight on the many-to-many relationships between the clusters of the two clusterings.

Figure 1 Comparing two clusterings of the same 2D data set involving 40 points. **(a)** Clustering F contains 2 clusters of respectively 30 and 10 points. Clustering F' contains 5 clusters of respectively 5, 15, 5, 5, and 10 points. In (b) and (c), the intersection graph associated with the two clusterings is depicted: one node per cluster, an edge between two nodes if the corresponding clusters share at least one point, the weight of an edge being the number of points shared by the two clusters. Our method groups clusters within meta-clusters. It is parameterized by the diameter D of the sub-graphs connecting clusters within meta-clusters (in red). Existing methods based on (maximum) graph matching correspond to $D = 1$. **(b)** With $D = 1$, a matching is obtained: F_1 with F'_2 and F_2 with F'_5. **(c)** With $D = 2$, $\{F_1\}$ is matched with the meta-cluster involving $\{F'_1, F'_2, F'_3, F'_4\}$, while $\{F_2\}$ is matched with $\{F'_5\}$.



In *information theoretical methods*, the mutual information (MI) between the two clusterings (i.e. the Kullback-Leibler divergence between the joint and the marginal distributions associated with the contingency table) is first computed. In combination with the entropy of marginal distributions of clusters' sizes, the MI can then used to derive the *Normalized mutual information* [44]. Alternatively, the MI can be used to define the variation of information (VI) between the clusterings F and F' [32]. In short, VI is defined from the mutual information between the two clusterings [8], namely the Kullback–Leibler divergence between the joint distribution and the marginals defined from the contingency table. While VI defines a metric, it exhibits the drawbacks of pair counting methods. Generalization of information theoretic measures have been proposed. In [45], the bipartite graph whose nodes are clusters with one weighted edge for two clusters sharing items is used to derive a general connected component-based decomposition formula, and it is shown that several existing measures are special cases. In a similar spirit, a generalization of clustering agreement measures is proposed in [36] based on a function $\varphi(\cdot)$ which quantifies the dispersion of the per-row and per-column distributions of the contingency table. Adjustments of mutual information measures have also been worked out in particular to reduce selection bias [38].

Optimal transportation based methods [48] aim at mapping the clusters with one another, and also at accommodating the case of soft clustering. These methods actually rely on the earth mover distance [39]–a linear program which may be seen as a particular case of optimal transportation. In short, this LP involves the distances between the clusters representatives (centroids), and solves for the weight assigned to the match between any two clusters. This approach is powerful, as fractional cluster matching goes beyond the 1-to-1 greedy matching alluded to above. However, the involvement of cluster centroids masks individual contributions from the items themselves, so that the approach does not apply when commonalities between groups of clusters from F and F' are sought.

Naturally, when the two clusterings studied stem from two runs of the same algorithm (randomized or with different initial conditions), the previous quantities can be used to assess the

stability of this algorithm [30]. In particular, the minimal matching distance has been used to study the stability of k-means.

Finally, the problem of computing many-to-many correspondences between two sets of clusters $S \subset F$ and $S' \subset F'$ was addressed recently [20]. The quality of the match is assessed by size of the symmetric difference of the union of clusters i.e. $\phi(S, S') = \mid \cup_{s \in S} s \ \Delta \ \cup_{s' \in S'} s \mid$, which ought to be minimized. The authors introduce four types of constraints, that denoted C_P stating that $S \notin \{\emptyset, F\}$. Under C_p, S defines a cut of the partition F, and the size of the cut is defined by $\phi_{F'}(S) = \min_{S' \subset F'} \phi(S, S')$. For a given S, the optimal partner S' can be found. For the max-flow min cut problem, recall that the Gomory-Hu construction guarantees that any graph admits a flow equivalent tree whose paths define the min cuts for all pairs of vertices. Analogously to graphs, there exists a min cut basis of F wrt $\phi_{F'}(\cdot)$, of size $|F'| - 1$. The computation of such cuts falls in the realm of submodular optimization and with exact algorithms which are essentially quadratic in the size of the partitions. A heuristic combining branch-and-bound and greedy techniques is also proposed.

Estimating the number of clusters. Finding the *correct* number of clusters k is admittedly a difficult problem since there is no universal definition of cluster [42], and also since the *right* number of clusters in a dataset often depends on the *scale* at which the dataset is inspected [41]. Nevertheless, various methods were developed [42, 28]. In the sequel, we consider in turn specific and generic methods.

For `k-means`, in order to deal with clusters with complex shapes and/or noise, the split-and-merge procedure [33, 1] first over-segmentates clusters (to ensure that each cluster center of mass is located in *high* density area), and further merges neighboring clusters. Clustering methods based on Gaussian mixtures triggered specific developments to accommodate clusters with non Gaussian shapes. The integrated completed likelihood (ICL) method [3] combines the Bayesian information criterion (BIC) and an entropic penalty on the model complexity. In a nearby vein, various hierarchical merging strategies of Gaussian components were investigated [23].

Generic methods may be classified depending on whether they directly operate on the clusters, or use a function (density estimate) modeling the underlying distribution of items/samples. Consider first methods in the first class. The *elbow* method tracks an abrupt change of the functional assessing the clustering quality [34]. The *silhouette* method [26] picks the clustering maximizing the silhouette of each sample–a number in $-1 \dots 1$ assessing whether the sample is well clustered. Under appropriate *well separated-ness* hypothesis, the *gap statistic* defines k by maximizing the change in within-cluster dispersion that is expected under an appropriate reference null distribution [42]. Consider next methods in the second class. In spirit, these methods *directly* read the correct number of clusters from connected components and/or local maxima of a density estimate defined from the samples. This idea was first implemented by tracking the connected components of a union of balls centered on the samples [9], a strategy akin to density estimation, a difficult problem in high dimensional spaces. More recently, it was shown that topological persistence is instrumental to read pertinent scales for local maxima, when such scales exist [6].

1.2 Exploiting many-to-many correspondences: illustration on a specific application

Having commented on the comparison of clusterings, let us address the status of individual items identified thanks to many-to-many correspondences between clusters. While most methods

summarize the comparison of the two clusterings with a real value, in a number of applications, though, the mapping between individual items from the two clusterings is critical.

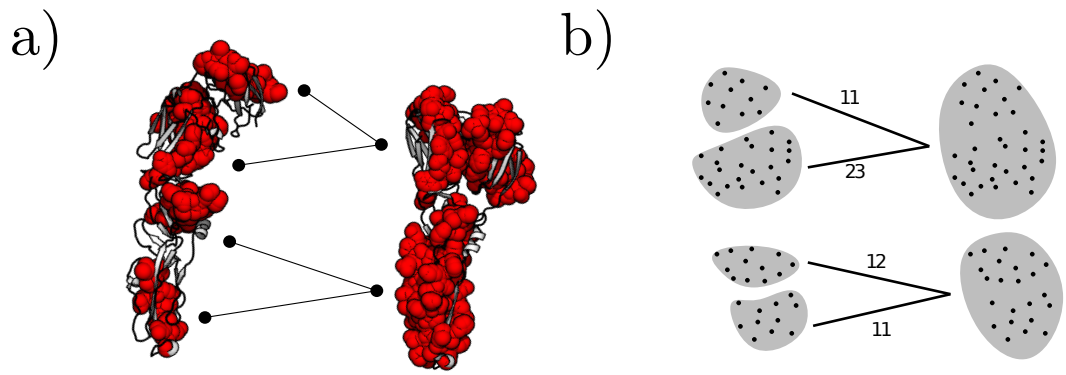
Our interest in the cluster comparison problem actually comes from the problem of comparing two protein molecules in structural biology. To apprehend the importance of this task recall that it is the structure (and also the dynamics) of molecules which determine their function. Prosaically, the structure of a lock must match that of the lock to be opened. In analyzing two protein conformations, a central task is to identify regions within these conformations, called structural motifs, which can be geometrically superimposed onto one another. Using geometric methods akin to rigidity analysis, a method to identify geometrically conserved regions was proposed in [4]. Once such regions have been identified, a many-to-many correspondence is sought (Fig. 2). Critically, one does not know a priori how many structural motifs a protein contains. Therefore, it is important not derive the correspondence sough from a cut as in [20]. Once identified, motifs can be used to boost the exploration of the conformational variability of a molecule. Since motifs are more rigid than their complement in the molecule, one can e.g. rigidify all degree of freedoms of atoms present in the motifs, restricting conformational changes to linker regions connecting motifs. This strategy is currently being exploited for several complex biomolecular systems such as membrane transporters (involved in cancer chemo-resistance and antibiotics resistance), the polymerase of influenza (which replicates the viral genome), as well as viral fusion proteins (used by viruses to attach themselves to the to-be-infected cells). Such a system has $3n$ Cartesian coordinates for n atoms. With typically $n > 10,000$, the brute force exploration of the conformation space requires massive calculations. With the detection of motifs and their complement (flexible linkers), the number of degrees of freedom to be considered is typically lowered by two orders of magnitude on the aforementioned systems – a dramatic simplification for the physical simulation.

1.3 Main contributions

Rationale. Until recently, previous work has overlooked two issues. First, the comparison of two clusterings has been done globally, i.e. without providing insights on the mapping between clusters–if one omits the elementary matching case. Second, in assessing and comparing clusterings, the *scale* at which clusters merge has not been studied. Phrased differently and following [41]: "*In fact, the* `right` *number of clusters in a dataset often depends on the* `scale` *at which the dataset is inspected*". Indeed, VI or co-occurrence matrices which count the number of times two data points are clustered together do not provide such insights. The recent contribution [20] partly answers the questions of which clusters match to which, in a global manner since binary cuts of the two clusterings / partitions are returned.

In this paper, we go beyond this approach by studying the problem of grouping clusters into meta-clusters, beyond cuts [20]. To do so, we define the family-matching problem on the intersection graph G constructed from F and F'. A node of G represents a cluster, an edge between two nodes means that the intersection between the two corresponding clusters is not empty, and the weight of an edge is the number of items (elements) shared by the two clusters (that necessarily belong to different clusterings). Intuitively, the family-matching problem consists in finding an explicit many-to-many correspondence between groups of clusters of the two clusterings. More formally, the family-matching problem consists in computing disjoint subsets of nodes (clusters of clusters, or meta-clusters) such that **(i)** every such subset induces a sub-graph of G of diameter at most a given constant $D \geq 1$, and **(ii)** the number of items, for which the two clusters that contain it (in F and in F') are in a same meta-cluster, is maximum. This parameter corresponds to the score of a solution. In this paper, we consider the unweighted diameter of graphs (largest number of edges which must be traversed in order to travel from one vertex to another or infinite

Figure 2 Finding structural motifs in protein conformations using clusters of clusters. **(a)** Two protein conformations reconstructed by X-ray crystallography, each with a set of amino-acids identified from a geometric analysis seeking (quasi-)isometric regions. On this simple example, regions from the left hand side are either nested or intersect with the motifs on the left (w.r.t their constituting amino-acids). Intersections are depicted as an edge. Structural motifs are defined from a many-to-many correspondence between these regions. The motifs identified this way are made rigid, which allows focusing the conformational exploration on flexible linkers. The process typically results in a two-fold reduction of the number of degrees of freedom to consider for the physical simulation. **(b)** Specification of the many-to-many correspondence problem, with one cluster for each rigid region, and one point for each amino-acid. Two clusterings of the same dataset. The value decorating an edge is the number of amino acid shared by two regions.



if the graph is not connected).

The constraint on the diameter D actually sheds light on previous work. The case $D = 1$ corresponds to previous work focused on 1-1 matchings. The case for which one cluster of F (F', respectively) corresponds to different smaller clusters of F' (F, respectively) is contained in the case where $D = 2$. We call this case 1-to-many. We prove in Lemma 6 that the optimal score for $D = 2$ can be arbitrarily large compared to the optimal score for $D = 1$ – an incentive to introduce this diameter constraint. The case $D > 2$ (constant) deals with the case where different clusters of F correspond to different clusters of F' (and vice versa) but without a *good* matching between these clusters. In that case, the value of D is a measure of the complexity of the two clusters of clusters (the meta-cluster involving these two). More generally: the case 2-to-many may be solved with $D = 4$, the case d-to-many may be solved with $D = 2d$. We do not have an equivalence because additional conditions are needed and, in practice, different values of D must be chosen because of the different scales of the clusterings.

On general (*i.e.* not necessarily bipartite) graphs, the D-family-matching problem is strongly related to the so-called *Maximum Edge Clique Partition* problem, in which one is looking for a partition of the vertices of an unweighted graph into cliques, maximizing the number of edges within the cliques. More precisely, Maximum Edge Clique Partition can be seen as the 1-family-matching problem in general graphs. As observed by [12], in an optimal solution of Maximum Edge Clique Partition, the maximum size of the clusters is not far from the size of a maximum clique of the input graph, which can be used in order to show that this problem cannot be approximated within $n^{1-\epsilon}$ for any ϵ unless $P = NP$ [31]. This approach could be used in

order to show that the 2-family-matching problem is not approximable within any ratio by using *Maximum Edge Biclique* as starting problem, in which the aim is to find a complete bipartite graph with the maximum number of edges in a bipartite graph. Indeed, Maximum Edge Biclique was recently shown to be $n^{1-\epsilon}$-inapproximable unless $P = NP$ for any $\epsilon > 0$ [31]. Unfortunately, in the bipartite case, an optimal solution of 2-family-matching may contain only small bicliques compared to an optimal solution of Maximum Edge Biclique: consider a star with q branches, and add two leaves to each branch. Observe that an optimal solution of 2-family-matching in this graph consists of the disjoint union of q paths of length 3 (thus, the maximum biclique in this solution consists of 2 edges only), while there exists a biclique with q edges.

Finally, our problem is also close to some classical partitioning problems. Indeed, separating an edge weighted graph (with non negative weights) into k components while minimizing the size of the cut is a classical optimization problem, known as *Minimum k-Cut* or *Minimum k-Way Cut* [19]. This problem is NP-hard if k is an input parameter but polynomial-time solvable for every fixed k [21]. Moreover, there exist polynomial-time algorithms with an approximation ratio better than 2 [40, 47], and a Fixed-Parameter Tractable (FPT) algorithm when parameterized by the number of edges in the cut [27], while it is W[1]-hard when parameterized by k [14]. On the other hand, when we seek for a minimum cut separating k prescribed vertices, the problem is already NP-hard for $k = 3$, but admits a $(2 - 2/k)$-approximation algorithm for every $k \geq 2$ [11]. The main difference between these approaches and our application lies in the specification of k, since we do not know a priori how many meta-clusters are sought. Admittedly, we also face a model selection problem for the choice of D, which is the focus of Section 6.

Contributions. Our work, which investigates the relationship between two clusterings, shedding light on the way clusters from one have been merged / split / edited to define one clustering from the other, consists of the following contributions. In Section 2, we introduce a new combinatorial optimization problem on the intersection graph, namely the D-family-matching problem, so as to compare two clusterings. In Section 3, we prove that the problem is very hard to solve: NP-completeness and APX-hardness results, and unbounded approximation ratio of simple strategies. In Section 4, we design exact polynomial time dynamic programming algorithms for some classes of instances (trees, paths, cycles, graphs of maximum degree two). In Section 5, we describe efficient algorithms for general graphs, introducing a variant of the problem with spanning tree constraints. In Section 7, we illustrate the ability of our algorithms to identify relevant meta-clusters between a given clustering and an edited version of it, and compare our scores against the Variation of Information. In both cases, we show that parameter D yields insights on the *scale* at which clusters coalesce.

2 Comparison of clusterings: formalization as graph problems

In this section, we formalize the D-family-matching problem modeling the comparison of clusterings. Let $t \geq 1$ be any positive integer. Let us consider a set of elements $Z = \{z_1, \ldots, z_t\}$. We are given two different families F and F' of disjoint subsets of Z. Let $r \geq 1$ be the size of F. Formally $F = \{F_1, \ldots, F_r\}$, where $F_i \subseteq Z$, $F_i \neq \emptyset$, and $F_i \cap F_j = \emptyset$ for every $i, j \in \{1, \ldots, r\}$, $i \neq j$. Let $r' \geq 1$ be the size of F'. In a analogous way, $F' = \{F'_1, \ldots, F'_{r'}\}$, where $F'_i \subseteq Z$, $F'_i \neq \emptyset$, and $F'_i \cap F'_j = \emptyset$ for every $i, j \in \{1, \ldots, r'\}$, $i \neq j$.

Definition. 1 (Edge-weighted intersection graph)**.** *The edge-weighted intersection graph $G = (U, U', E, w)$ associated with Z, F, and F', is constructed as follows. The set $U = \{u_1, \ldots, u_r\}$ corresponds to the clustering F. To each vertex u_i, we associate the set $F_i \in F$. The set*

$U' = \{u'_1, \ldots, u'_{r'}\}$ corresponds to the clustering F'. To each vertex u'_i, we associate the set $F'_i \in F$. The set of edges of G is $E = \{\{u_i, u'_j\} \mid F_i \cap F'_j \neq \emptyset, 1 \leq i \leq r, 1 \leq j \leq r'\}$. The weight of any edge $e = \{u_i, u'_j\} \in E$ is $w_e = |F_i \cap F'_j|$.

In the rest of the paper we will write intersection graph instead of edge-weighted intersection graph and $w_{u,u'}$ instead of $w_{\{u,u'\}}$ to denote the weight of an edge $\{u, u'\} \in E$. See Figure 1 and Figure 3 for two detailed examples. We prove in Lemma 1 that any edge-weighted bipartite graph G with positive integers, is an intersection graph for some Z, F, and F'. A bipartite graph is a graph whose nodes can be partitioned into two disjoint sets such that every edge has an extremity in the first set and has its other extremity in the second set.

Lemma. 1. *Let $G = (V, E, w)$ be any edge-weighted bipartite graph such that $w_e \in \mathbb{N}^+$ for every $e \in E$. Then, there exist Z, F, and F' for which G is the intersection graph.*

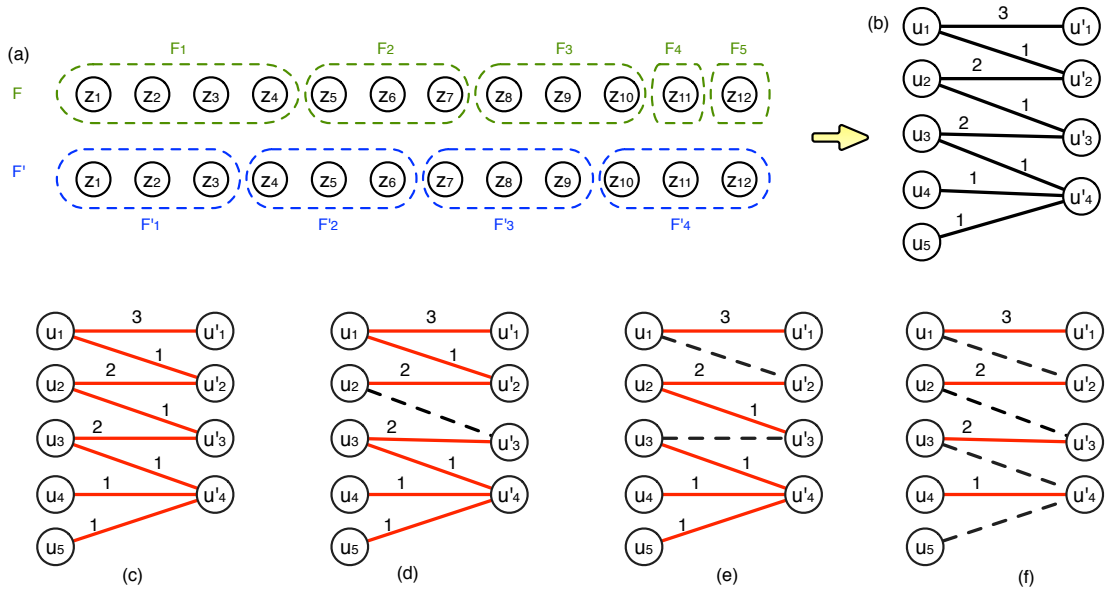
Proof of Lemma 1. Without loss of generality, we assume that G is connected (otherwise, we prove the result for every maximal connected component). We prove the result by induction on the number of nodes n. Let $V = U \cup U'$. Consider first that $n = |U \cup U'| = 2$. Let $U = \{u_1\}$ and $U' = \{u'_1\}$. We construct Z, F, and F' as follows. Set $Z = \{z_1, \ldots, z_t\}$ with $t = w_{u_1,u'_1}$. Set $F = \{F_1\}$ with $F_1 = \{z_1, \ldots, z_t\}$ and set $F' = \{F'_1\}$ with $F'_1 = \{z_1, \ldots, z_t\}$. Thus, G is the intersection graph for Z, F, and F'.

Suppose now that it is true for every edge-weighted bipartite graph composed of at most n nodes and such that the weights are positive integers. We prove that it is also true for every edge-weighted bipartite graph $G = (U, U', E, w)$ such that $|U \cup U'| = n + 1$ and such that the weights are positive integers. Consider a node $x \in U \cup U'$ such that $G' = G[(U \cup U') \setminus \{x\}]$ is connected. By induction hypothesis, G' is an intersection graph. We define $Z^{G'}$, $F^{G'}$, and $F'^{G'}$ corresponding to G' as follows. Let $Z^{G'} = \{z_1, \ldots, z_t\}$, $F^{G'} = \{F_1, \ldots, F_r\}$, and $F'^{G'} = \{F'_1, \ldots, F'_{r'}\}$. Without loss of generality, assume that $x \in U$. Let $N_G(x) = \{u'_1, \ldots, u'_{d_x}\}$, where d_x is the number of neighbors of x in G. Without loss of generality, assume that u'_i corresponds to F'_i for every $i \in \{1, \ldots, d_x\}$ (we permute the indices otherwise). Set $w_x = \sum_{i=1}^{d_x} w_{x,u'_i}$. We construct Z, F, and F' corresponding to G as follows. Set $Z = Z^{G'} \cup \{z_{t+1}, \ldots, z_{t+w_x}\} = \{z_1, \ldots, z_t, z_{t+1}, \ldots, z_{t+w_x}\}$. Set $F = \{F_1, \ldots, F_r, F_{r+1}\}$, where $F_{r+1} = \{z_{t+1}, \ldots, z_{t+w_x}\}$. For every $i, j \in \{1, \ldots, d_x\}$, $i \neq j$, let $X_i \subseteq \{z_{t+1}, \ldots, z_{t+w_x}\}$ with $|X_i| = w_{x,u'_i}$ and such that $X_i \cap X_j = \emptyset$. Finally, set $F' = \{F''_1, \ldots, F''_{r'}\}$, where $F''_i = F'_i \cup X_i$ for every $i \in \{1, \ldots, d_x\}$, and $F''_i = F'_i$ for every $i \in \{d_x + 1, \ldots, r'\}$. We get that G is the intersection graph for Z, F, and F'. Thus, the result is true for every edge-weighted bipartite graph $G = (U, U', E, w)$ such that $2 \leq |U \cup U'| \leq n + 1$ and such that the weights are integers. □

By Lemma 1, we can focus on any intersection graph without necessarily considering the corresponding Z, F, and F'. In the rest of the paper, an intersection graph will be denoted $G = (V, E, w)$. Let us define some notations. We denote by $n = |V|$ the number of nodes of G, by $m = |E|$ the number of edges of G, and by $\Delta = \max_{v \in V} |N_G(v)|$ the maximum degree of G, where $N_G(v)$ is the set of neighbors of $v \in V$ in G. The diameter of a graph is the maximum number of edges of a shortest path in this graph. The set $cc(G)$ represents the set of maximal connected components of G. We now define the notion of D-family-matching.

Definition. 2 (D-family-matching)**.** *Let $D \in \mathbb{N}^+$. Let $G = (V, E, w)$ be an intersection graph. A D-family-matching for G is a family $\mathcal{S} = \{S_1, \ldots, S_k\}$, $k \geq 1$, such that, for every $i, j \in \{1, \ldots, k\}$, if $i \neq j$, then: $S_i \subseteq V$, $S_i \neq \emptyset$, $S_i \cap S_j = \emptyset$, and the graph $G[S_i]$ induced by the set of nodes S_i has diameter at most D.*

Figure 3 **Simple instance of the D-family-matching problem and solutions: panels (c,d,e,f) represent optimal solutions for different values of D. (a)** Simple instance of the D-family-matching problem with $t = 12$, $r = 5$, $r' = 4$, and so $n = 9$. The family F contains five sets and the family F' contains four sets. **(b)** Intersection graph G. **(c)** Optimal solution $\mathcal{S}$ for $D \geq 7$ with $\Phi(\mathcal{S}) = \Phi_D(G) = 12$. **(d)** Optimal solution $\mathcal{S}$ for $D = 3$ with $\Phi(\mathcal{S}) = \Phi_3(G) = 11$. **(e)** Optimal solution $\mathcal{S}$ for $D = 2$ with $\Phi(\mathcal{S}) = \Phi_2(G) = 9$. Observe that there is another optimal solution by removing the two edges $\{u_2, u'_3\}$ and $\{u_3, u'_4\}$ and by adding the edge $\{u_3, u'_3\}$. **(f)** Optimal solution $\mathcal{S}$ for $D = 1$ with $\Phi(\mathcal{S}) = \Phi_1(G) = 8$.



The score $\Phi(\mathcal{S})$ of a D-family-matching $\mathcal{S}$ is $\Phi(\mathcal{S}) = \sum_{i=1}^{k} \sum_{e \in E(G[S_i])} w_e$. Let $\mathcal{S}_D(G)$ be the set of all D-family-matching for G. We now formalize the D-family-matching problem. Intuitively, we wish to compute a D-family-matching which minimizes the inconsistencies.

Definition. 3 (D-family-matching problem)**.** *Let $D \in \mathbb{N}^+$. Given an intersection graph G, the D-family-matching problem consists in computing $\Phi_D(G) = \max_{\mathcal{S} \in \mathcal{S}_D(G)} \Phi(\mathcal{S})$.*

From such an optimal solution, we deduce an optimal number of sets $k \geq 1$ for our clusterings comparison. Observe that the 1-family-matching problem is equivalent to the problem of computing a maximum weighted matching in weighted bipartite graphs. Since this problem can be solved in $O(n^2 \log n + nm)$ [18], we deduce the following result.

Lemma. 2. *Given any intersection graph G, the 1-family-matching problem can be solved in $O(n^2 \log n + nm)$.*

Figure 3 illustrates the problem and we summarize the notations in Table 1.

2.1 Equivalent definition of the D-family-matching problem

We now define an equivalent definition of the D-family-matching.

Definition. 4 (D-family-matching)**.** *Let $D \in \mathbb{N}^+$. A D-family-matching is a family $\mathcal{P} = \{P_1, \ldots, P_k\}$, $k \geq 1$, of subsets of $F \cup F' = \{F_1, \ldots, F_r, F'_1, \ldots, F'_{r'}\}$ such that, for every*

Table 1 Notations

Notation	Definition
$Z = \{z_1, \ldots, z_t\}$	Set of $t \geq 1$ elements
$F = \{F_1, \ldots, F_r\}$	Family of $r \geq 1$ disjoint subsets of Z
$F' = \{F'_1, \ldots, F'_{r'}\}$	Family of $r' \geq 1$ disjoint subsets of Z
$G = (V, E, w)$	Intersection graph of $n \geq 1$ nodes and $m \geq 1$ edges
$N_G(v) = \{v' \mid \{v, v'\} \in E\}$	Set of neighbors of node $v \in V$
$\Delta = \max_{v \in V} \lvert N_G(v) \rvert$	Maximum degree of G
$cc(G)$	Set of maximal connected components of G
$\mathcal{S} = \{S_1, \ldots, S_k\}$	D-family-matching
$\Phi(\mathcal{S}) = \sum_{i=1}^{k} \sum_{e \in E(G[S_i])} w_e$	Score of a D-family-matching $\mathcal{S}$
$\mathcal{S}_D(G)$	Set of all D-family-matching for G
$\Phi_D(G) = \max_{\mathcal{S} \in \mathcal{S}(G,D)} \Phi(\mathcal{S})$	Optimal score for the D-family-matching problem
$\mathcal{S}_D(G, T_r)$	Set of all D-family-matching constrained by T_r
$\Phi_D(G, T_r) = \max_{\mathcal{S} \in \mathcal{S}_D(G,T_r)} \Phi(\mathcal{S})$	Optimal score for the D-family-matching problem constrained by T_r

$i, j \in \{1, \ldots, k\}$, *$i \neq j$, then:* $P_i \subseteq F \cup F'$, $P_i \neq \emptyset$, $P_i \cap P_j = \emptyset$, *and* $\mathcal{P}$ *must satisfy the* diameter constraints*: for every* $H, H' \in P_i$, *then there exists a sequence* $(H_0, \ldots, H_d)$ *such that* $d \leq D$, $H_0 = H$, $H_d = H'$, $H_j \in P_i$, *and* $H_j \cap H_{j+1} \neq \emptyset$ *for every* $j \in \{0, \ldots, d-1\}$.

The score $f(\mathcal{P})$ of a D-family-matching $\mathcal{P}$ is defined as follows:

$$f(\mathcal{P}) = \sum_{i=1}^{k} |(P_i \cap_F F) \cap_Z (P_i \cap_{F'} F')|.$$

Let $\mathcal{P}_D(F, F')$ be the set of all D-family-matching for F, F', and D. We now formalize an equivalent definition of the D-family-matching problem.

Definition. 5 (D-family-matching problem)**.** *Let* $D \in \mathbb{N}^+$. *The D-family-matching problem consists in determining a D-family-matching that maximizes the score f. Formally, we aim at computing:*

$$f_D(F, F') = \max_{\mathcal{P} \in \mathcal{P}_D(F,F')} f(\mathcal{P}).$$

Finally, we obtain the following property showing the equivalence between the two definitions of the D-family-matching problem.

Property 1. *Let* $D \in \mathbb{N}^+$. *Let* $L \geq 0$ *be any positive real number. Consider any instance of the D-family-matching problem defined by Z, F, and F', and consider the associated intersection graph G. Then, there is a D-family-matching $\mathcal{P}$ for Z, F, and F', such that $f(\mathcal{P}) \geq L$ if and only if there is a D-family-matching $\mathcal{S}$ of G such that $\Phi(\mathcal{S}) \geq L$.*

3 Hardness of the D-family-matching problem and greedy strategies

We prove that the D-family-matching problem is APX-hard (Section 3.1) and that simple strategies can be arbitrarily bad (Section 3.2).

3.1 APX-hardness

As explained before, the 1-family-matching problem is polynomial-time solvable, thus we now focus on higher values of D. Moreover, we will prove in Section 4 that the problem is polynomial-time solvable for bipartite graphs of maximum degree $\Delta = 2$. We prove that these two cases are actually the only pairs (D, Δ) leading to polynomial problems, all other being NP-complete and even APX-hard. We were also able to prove hardness results for some (D, Δ) even in the case where edge weights are within a fixed range of values.

Theorem. 2. *Let $D \geq 2$ be any integer. The decision version of the D-family-matching problem is APX-hard for :*

- *bipartite graphs of maximum degree 3;*
- *bipartite graphs of maximum degree 4 when the maximum weight is constant.*

Moreover, the 2-family-matching problem is APX-hard for bipartite graphs of maximum degree 3 with unary weights.

Although we leave as an open question whether D-family-matching is in APX (namely, whether it admits a polynomial-time algorithm achieving a constant approximation ratio), we show that two natural strategies for obtaining approximation algorithms are hopeless.

The rest of this section is devoted to the proof of Theorem 2. For the sake of readability, we splitted this proof into three parts: Theorems 3, 7 and 8. Notice that the last two proofs are quite similar.

Let us first recall the definition of L-reduction in order to transfer approximation lower bounds.

Definition. 6. *[35] Let Π and Π' be two maximization problems. We say that Π L-reduces to Π' is there are two polynomial-time algorithms f, g and constants $\alpha, \beta > 0$ such that for each instance I of Π:*

1. *Algorithm f produces an instance $I' = f(I)$ of Π' such that the optima of I and I', denoted by $OPT_\Pi(I)$ and $OPT_{\Pi'}(I')$, respectively, satisfy $OPT_{\Pi'}(I') \leq OPT_\Pi(I)$;*
2. *Given any solution of I' with cost c', algorithm g produces a solution of I with cost c such that $OPT_\Pi(I) - c \leq \beta(OPT_{\Pi'}(I') - c')$.*

It is known that if Π is APX-hard and L-reduces to Π', then Π' is APX-hard as well. In that case, Π' does not admit a $PTAS$ (Polynomial Time Approximation Scheme) unless $P = NP$.

3.1.1 Part I: $\Delta = 4$, fixed values of weights

Theorem. 3. *Let $D \geq 2$ be a positive integer. The D-family-matching problem is APX-hard even if the maximum degree Δ is at most 4 and the number of different weights is at most 3.*

In our reduction, we use a special case of set packing problem, a well known NP-complete problem [25]. Given a universe $X = \{x_1, \ldots, x_t\}$ of $t \geq 1$ elements and a family $Y = \{Y_1, \ldots, Y_p\}$ of $p \geq 1$ subsets of X, a *packing* is a subfamily $\mathcal{C} \subseteq Y$ of subsets such that all set in $\mathcal{C}$ are pairwise disjoint, that is $Y_i \cap Y_j = \emptyset$ for all $Y_i, Y_j \in \mathcal{C}$, $i \neq j$. Given X, Y, and an integer $k \geq 1$, the *Set packing problem* consists in determining whether there exists a packing $\mathcal{C}$ of size $|\mathcal{C}| = k$. Notice that this problem remains APX-hard even if $|Y_i| = 3$ for every $i \in \{1, \ldots, p\}$ and x_i is in at most 3 sets, for every $i \in \{1, \ldots, t\}$ [24].

Consider any instance $\mathcal{I}_{sp}$ of set packing problem: a universe $X = \{x_1, \ldots, x_t\}$, a family $Y = \{Y_1, \ldots, Y_p\}$ of subsets of X, and an integer $k \geq 1$. We assume that $|Y_i| = 3$ for all $i \in \{1, \ldots, p\}$ and that x_i is in at most 3 sets. We first construct the intersection graph G of the D-family-matching problem (Definition 7).

Definition. 7 (Construction of the intersection graph G for the D-family-matching problem)**.** *The intersection graph $G = (V, E, w)$ is defined as follows.*

- *Set $V = U \cup U' \cup Z$, where*
  - *$U = \{u^1, \ldots, u^p\}$ corresponds to Y,*
  - *$U' = \{u'_1, \ldots, u'_t\}$ corresponds to X,*
  - *and $Z = \{z^i_j, 1 \leq i \leq p, 1 \leq j \leq D\}$.*
- *Set $E = E_U \cup E_Z \cup E_Z$, where*
  - *$E_U = \{\{u^i, u'_j\} \mid x_j \in Y_i, 1 \leq i \leq p, 1 \leq j \leq t\}$,*
  - *$E_Z = \{\{z^i_j, z^i_{j+1}\} \mid 1 \leq i \leq p, 1 \leq j \leq D-1\}$,*
  - *and $E_{UZ} = \{\{u^i, z^i_1\} \mid 1 \leq i \leq p\}$.*
- *Set $w_e = 2$ for every $e \in E_U$, $w_{\{z^i_{D-1}, z^i_D\}} = 5$ for every $i \in \{1, \ldots, p\}$, and $w(e) = T$ for every $e \in E_{UZ} \cup E_Z \setminus \{\{z^i_{D-1}, z^i_D\} \mid 1 \leq i \leq p\}$, where*

$$T = 9(2^D - 1)^2 + 3(2^D - 1) + 6.$$

Observe that the maximum degree Δ of G is at most 4 and that G is bipartite.

Intuitively, we will prove that any optimal solution $\mathcal{S}$ has size p and for every $S \in \mathcal{S}$, S contains exactly $D-1$ edges of weight T plus either one edge of weight 5 or three edges of weight 2 (such case will correspond to the subsets of the set packing).

Lemma. 3. *If there is a solution $\mathcal{C}$ for the instance $\mathcal{I}_{sp}$ of set packing problem such that $|\mathcal{C}| \geq k$, then there is a solution $\mathcal{S}$ for the D-family-matching problem for G such that $\Phi(\mathcal{S}) \geq p(D-1)T + 5p + k$.*

Proof of Lemma 3. Consider any solution $\mathcal{C}$ for the instance $\mathcal{I}_{sp}$ of set packing problem such that $|\mathcal{C}| = k$. We construct a solution $\mathcal{S}$ for the D-family-matching problem for G such that $\Phi(\mathcal{S}) = n(D-1)T + 5n + k$. Assume that $\mathcal{C} = \{Y_1, \ldots, Y_k\}$ (we permute the indices otherwise). Let $\mathcal{S} = \{S_1, \ldots, S_p\}$, where $S_i = \{u^i\} \cup N_G(u^i) \cup Z^i \setminus \{z^i_D\}$ for every $i \in \{1, \ldots, k\}$ and $S_i = Z^i \cup \{u^i\}$ for every $i \in \{k+1, \ldots, p\}$. The sets are disjoint. In other words, for every $i, j \in \{1, \ldots, p\}$, $i \neq j$, then $S_i \cap S_j = \emptyset$ because $\mathcal{C}$ is a set packing and, by construction of G, we have $N_G(u^{i'}) \cap N_G(u^{j'}) = \emptyset$ for every $i', j' \in \{1, \ldots, k\}$, $i' \neq j'$. Furthermore, for every $i \in \{1, \ldots, p\}$, the diameter of $G[S_i]$ is at most D. Finally, we get

$$\Phi(\mathcal{S}) = \Phi(\{S_1, \ldots, S_k\}) + \Phi(\{S_{k+1}, \ldots, S_p\}) = k[(D-1)T+6] + (p-k)[(D-1)T+5] = p(D-1)T+5p+k.$$

Thus, we have proved that $\mathcal{S}$ is a solution for the D-family-matching problem for G such that $\Phi(\mathcal{S}) = p(D-1)T + 5p + k$. □

Lemma. 4. *If there is a solution $\mathcal{S}$ for the D-family-matching problem for G such that $\Phi(\mathcal{S}) \geq p(D-1)T + 5p + k$, then there is a solution $\mathcal{C}$ for the instance $\mathcal{I}_{sp}$ of set packing problem such that $|\mathcal{C}| \geq k$.*

Proof of Lemma 4. Consider any optimal solution $\mathcal{S}$ for the D-family-matching problem. Without loss of generality, we assume that $\mathcal{S}$ contains the smallest number of sets. In other words, $|\mathcal{S}| \leq |\mathcal{S}'|$ for any solution $\mathcal{S}'$ such that $\Phi(\mathcal{S}') = \Phi(\mathcal{S})$. We deduce that every set of $\mathcal{S}$ contains at least two nodes. Otherwise, we can remove such single sets without decreasing the score. We first prove the following claims.

Claim 4. *Consider the graph G' induced by the set of nodes $U \cup U'$. Any subgraph of G' of diameter at most D is composed of at most $\frac{9}{2}(2^D-1)^2 + \frac{3}{2}(2^D-1)$ edges.*

Proof of Claim 4. Any graph of degree at most k and diameter at most D, is composed of at most $1 + k\sum_{i=0}^{D-1}(k-1)^i$ nodes (Moore bound). Recall that $|Y_i| = 3$ for all $i \in \{1,\dots,p\}$ and that x_i is in at most 3 sets. Thus, the graph G' has degree at most 3. We get that any subgraph of G' of diameter at most D is composed of at most $1 + 3\sum_{i=0}^{D-1} 2^i = 1 + 3(2^D-1)$ nodes. We deduce that the number of edges of such graphs is upper-bounded by $[1+3(2^D-1)][\frac{3}{2}(2^D-1)] = \frac{9}{2}(2^D-1)^2 + \frac{3}{2}(2^D-1)$. □

Claim 5. *For every $S \in \mathcal{S}$, then $|S \cap U| = 1$.*

Proof of Claim 5. We first prove that $|S \cap U| \leq 1$ for every $S \in \mathcal{S}$. By contradiction. Suppose that there exists $S \in \mathcal{S}$ such that $|S \cap U| \geq 2$. Without loss of generality, let $S \cap U = \{u_1,\dots,u_b\}$, $b \geq 2$. For every $i \in \{1,\dots,b\}$, then there exists $j \in \{1,\dots,D-1\}$ such that $z^i_j \notin S$. Indeed, the distance between z^i_{D-1} and $u_{i'}$ is at least $D+1$ for every $i, i' \in \{1,\dots,b\}$, $i' \neq i$. Intuitively, in that case the solution is missing at least one edge of weight T amongst the edges induced by $\{z^i_1,\dots,z^i_{D-1}\}$, for every $i \in \{1,\dots,\}$.

Consider the partition $\mathcal{S}'$ obtained from $\mathcal{S} \setminus S$ where we remove from every set all vertices $\{z^i_j, i \in \{1,\dots,b\}, j \in \{1,\dots,D\}\} \cup \{u^1,\dots,u^b\}$, and we add the sets $S^i = \{u^i, z^i_1,\dots,z^i_D\}$ for each $i \in \{1,\dots,b\}$. Observe that $\Phi(\mathcal{S}) - \Phi(\mathcal{S}') \geq bT - T > 0$ because, by Claim 4, the subgraph induced by the set of nodes $S \cap (U \cup U')$ has at most $\frac{9}{2}(2^D-1)^2 + \frac{3}{2}(2^D-1)$ edges. A contradiction because $\mathcal{S}$ is an optimal solution. Thus, for every $S \in \mathcal{S}$, we have $|S \cap U| \leq 1$.

Finally, suppose that there exists $S \in \mathcal{S}$ such that $|S \cap U| = 0$. We may assume that $S \cap U' = \emptyset$, for otherwise we would have $S = \{u'_j\}$ for some $j \in \{1,\dots,t\}$, but in that case $\mathcal{S} \setminus S$ would have the same weight as $\mathcal{S}$, and strictly less sets, which is impossible. Hence, there must exist $i \in \{1,\dots,p\}$ such that $S \subseteq \{z^i_1,\dots,z^i_D\}$. Consider the partition $\mathcal{S}'$ constructed from $\mathcal{S} \setminus S$, where we remove from each set the vertices $\{z^i_1,\dots,z^i_D\}$, and add the set $\{u^i, z^i_1,\dots,z^i_D\}$. Observe that $\Phi(\mathcal{S}') - \Phi(\mathcal{S}) \geq T$, a contradiction. □

Claim 6. *$\mathcal{S}$ contains exactly p sets.*

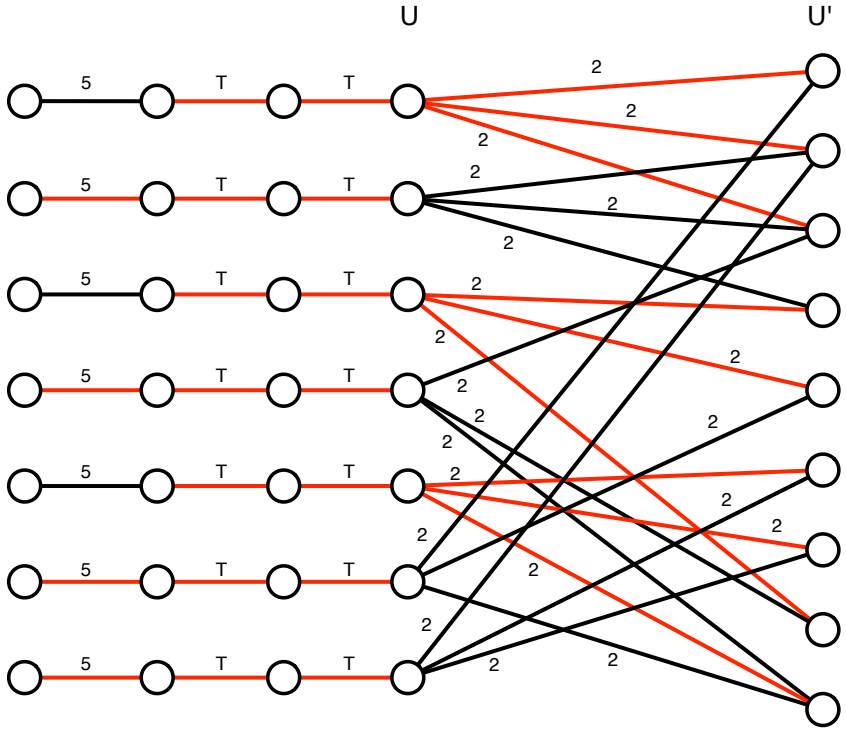
Proof of Claim 6. This holds by Claim 5 and the fact that every set of $\mathcal{S}$ must induce a subgraph of diameter at most D. In particular, every set must induce a connected graph. □

We are now able to prove Lemma 4. We assume that there is a solution $\mathcal{S}$ for the D-family-matching problem for G such that $\Phi(\mathcal{S}) \geq p(D-1)T + 5p + k$. Without loss of generality, we assume that $\mathcal{S}$ is optimal. Combining Claim 5 and Claim 6, we get that for every set $S \in \mathcal{S}$, there exists $i \in \{1,\dots,p\}$ such that either:

- $S = \{u^i, N_G(u^i), z^i_1,\dots,z^i_{D-1}\}$, in which case $\sum_{e \in E(G[S])} w_e = (D-1)T + 6$, or
- $S = \{u^i, z^i_1,\dots,z^i_D\}$, in which case $\sum_{e \in E(G[S])} w_e = (D-1)T + 5$.

Let us say that S is of type 1 (resp. type 2) if it falls in the first (resp. second) case. Since all sets of $\mathcal{S}$ are pairwise disjoint, they actually correspond to a solution $\mathcal{C}$ for the instance $\mathcal{I}_{sp}$. Since $\Phi(\mathcal{S}) \geq p(D-1)T + 5p + k$, we have that $|\mathcal{C}| \geq k$, as desired. □

Figure 4 Illustration of the proof of Theorem 3. See details in the text.



We are now able to prove the theorem.

Proof of Theorem 3. First, the reduction (Definition 7) can be clearly done in polynomial time. Finally, Lemmas 3 and 4 prove that any instance $\mathcal{I}_{sp}$ of set packing can be transformed into an instance G of D-family-matching, and any solution C of I of cost at least k can be transformed in polynomial time into a solution $\mathcal{S}$ of G of cost at least $p(D-1)T+5p+k$. We thus obtain the algorithms f and g of Definition 6. Now, observe that any instance I of set packing considered in our reduction is such that $|Y_i| = 3$ for every $i \in \{1,\dots,p\}$, and every element x_j belongs to at most three sets. This implies that any optimal solution $\mathcal{C}$ of I is such that $|\mathcal{C}| \geq \frac{p}{7}$. Indeed, the number of elements which belong to a set of $\mathcal{C}$ is exactly $3|\mathcal{C}|$, and any set must intersect one of these vertices. However, each of these vertices can belong to at most 2 sets outside $\mathcal{C}$. Hence $p \geq 7|\mathcal{C}|$.

These two arguments already prove item 1 of Definition 6. We then prove item 2 of Definition 6 with $\beta = 1$. Given a solution $\mathcal{C}$ of I obtained from a solution $\mathcal{S}$ of G, we have to show that $OPT_{SP}(I) - |\mathcal{C}| \leq \Phi_D(G) - |\mathcal{S}|$. However we have $\Phi_D(G) = OPT_{SP}(I) + p(D-1)T + 5p$, hence the required inequality is actually equivalent to $|\mathcal{C}| \geq |\mathcal{S}| - (p(D-1)T + 5p)$ which is true by the previous lemmas. □

To illustrate the proof of Theorem 3, consider the instance $\mathcal{I}_{sp}$ of set packing problem, where $X = \{x_1,\dots,x_9\}$, a family $Y = \{Y_1,\dots,Y_7\}$ of subsets of X such that $Y_1 = \{x_1,x_2,x_3\}$, $Y_2 = \{x_2,x_3,x_4\}$, $Y_3 = \{x_4,x_5,x_9\}$, $Y_4 = \{x_3,x_8,x_9\}$, $Y_5 = \{x_6,x_7,x_8\}$, $Y_6 = \{x_1,x_5,x_8\}$, $Y_7 = \{x_2,x_6,x_7\}$. Note that $p = 7$. Let $D = 3$. The graph G depicted in Figure 4 is the graph obtained from Definition 7. There is a set packing $\mathcal{C} = \{Y_1,Y_3,Y_5\}$ of size $k = 3$ and there is a

3-family-matching $\mathcal{S}$ such that $\Phi(\mathcal{S}) \geq p(D-1)T + 5p + k = 14T + 38 = 6590$ (depicted in red in Figure 4).

3.1.2 Part II: D = 2, Δ = 3, unary weights

We prove the following:

Theorem. 7. *The 2-family-matching problem is APX-hard in graphs of maximum degree 3 with unary weights.*

Proof. We reduce from the **Induced Matching Problem**, which takes as input a graph $G = (V, E)$ and an integer k, and asks whether there exists a set M of at least k edges such that no two edges of M are joined by an edge. Such a set M is called an *induced matching*. This problem is APX-complete on graphs on maximum degree 3 [10].

Let $G = (V, E)$ be a graph of maximum degree 3, with $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$. We construct a bipartite graph G' composed of the vertex set $V' \cup E'$, where $V' = \{v'_1, \ldots, v'_n\}$ and $E' = \{e'_1, \cdots e'_m\}$. Then, if $e_j = \{v_a, v_b\}$ is an edge of G, add to G' the edges $\{v'_a, e'_j\}$ and , $\{v'_b, e'_j\}$ and give them weight 1. We now prove that G contains an induced matching of size k if and only if G' contains a 2-family-matching of weight $m + k$.

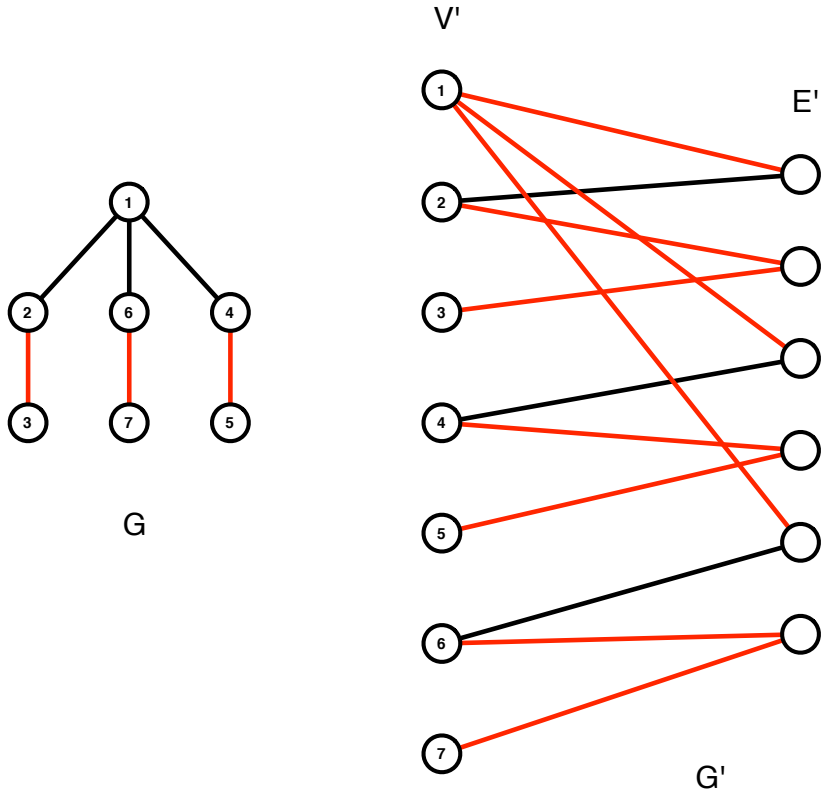
$\Rightarrow$ Let $M \subseteq E$ be an induced matching of G of size at least k. W.l.o.g., assume that $M = \{e_1, \ldots e_k\}$. Construct, for each $j \in \{1, \ldots, k\}$, the cluster $S_j^M = \{v'_a, v'_b, e'_j\}$, where a, b are such that $e_j = \{v_a, v_b\}$. Clearly S_j^M induces a graph of diameter 2. More precisely, it induces a path of length 2, its weight is thus 2. Moreover, there are k such clusters. Now, for every edge $e_j \in E \setminus M$, let i_j be such that $v_{i_j} \in e_j$ and v_{i_j} does not belong to any edge from M, chosen arbitrarily if several choices are possible.

Notice that v_{i_j} is well defined, since M is an induced matching. Let $\mathcal{I} = \{i_j | e_j \in E \setminus M\}$. For every $i \in \mathcal{I}$, construct the cluster $S_i^O = \{v'_i\} \cup \bigcup_{j|i_j=i} \{e'_j\}$. Here again, one can check that the diameter of the graph induced by S_i^O is 2, since it is a star whose center is v'_i, with either one, two or three branches, in which case its weight is respectively 1, 2 or 3. Moreover, the total weight of these clusters is $m - k$. Alltogether, we obtain a 2-family-matching of weight $2k + m - k = m + k$, as desired.

$\Leftarrow$ Conversely, let $\mathcal{S}$ be a solution of weight at least $m + k$. Observe first that since the vertices of E' are of degree 2, and none of them have the same neighborhood, each cluster must induce a star, centered at either a vertex from V' or E'. Let M be the set of all $j \in \{1, \ldots, m\}$ such that e'_j is in the same cluster as v'_a and v'_b, where $e_j = \{v_a, v_b\}$. Now, we show that we can modify the solution (without decreasing its weight) so that edges of index in M form an induced matching: suppose there exist $j, j' \in M$, $j \neq j'$, $r \notin M$ such that $e_r = \{v_a, v_b\}$ with $v_a \in e_j$ and $v_b \in e_{j'}$ (*i.e.* edges e_j and $e_{j'}$ are incident). Hence, e'_r is not contained in any cluster. We remove v'_a from the cluster which contains e'_j, and create a cluster $\{v'_a, e'_r\}$. Applying iteratively this modification we obtain a solution which is still a 2-family-matching of weight at least $k+m$. Moreover, edges whose index are in M form an induced matching. Finally, the number of edges contained in a cluster of $\mathcal{S}$ is at most $2|M| + |E \setminus M| = m + |M|$ and at least $m+k$, thus $|M| \geq k$.

In order to prove that this is indeed an L-reduction, it remains to show that any maximum induced matching M is of size at least αm for some fixed $\alpha > 0$. This is indeed the case, since the input graph has maximum degree 3. Hence, if M denotes a maximum induced matching and $V(M)$ denotes the vertices of M, observe that any edge of G is either (i) in M, (ii) not in M but induced by $V(M)$, (iii) incident to some vertex of $N(V(M))$, the neighbors of $V(M)$. Since $|V(M)| = 2|M|$, we have $|N(V(M))| \leq 6M$, and thus the total number of vertices is bounded above by $8M$, which gives a linear upper bound on m, as desired. □

Figure 5 Illustration of the proof of Theorem 7. See details in the text.



To illustrate the proof of Theorem 7 consider the instance of Induced Matching problem depicted in Figure 5 (left). Figure 5 (right) represents the intersection graph constructed from it. There is an induced matching of G of size 3 and there is a 2-family-matching composed of 4 sets with total weight equal to 9.

3.1.3 Part III: D > 2, Δ = 3

We prove the following result:

Theorem. 8. *For any $D \geq 3$, the D-family-matching problem is APX-hard for graphs of maximum degree 3.*

Proof. Similarly to the previous case, we reduce from the **Maximum Induced Matching Problem**.

Let $G = (V, E)$ be a graph of maximum degree 3, with $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$. We construct a bipartite graph G' composed of the vertex set $V^1 \cup E^1 \cup \cdots \cup E^{D-1} \cup V^2$, where $V^i = \{v_1^i, \ldots, v_n^i\}$ and $E^j = \{e_1^j, \cdots e_m^j\}$ for $i \in \{1, 2\}$, $j \in \{1, \ldots, D-1\}$. Then, if $e_j = \{v_a, v_b\}$ is an edge of G, add to G' the edges $\{v_a^1, e_j^1\}$, $\{v_b^1, e_j^1\}$, $\{e_j^{D-1}, v_a^2\}$, $\{e_j^{D-1}, v_b^2\}$ and give them weight 1. Finally, for every $j \in \{1, \ldots, m\}$ and $\ell \in \{1, \ldots, D-2\}$, add the edge $\{e_j^\ell, e_j^{\ell+1}\}$ and give it weight $W = 4m+1$. Clearly G' is a bipartite graph of maximum degree 3. We now prove that G contains an induced matching of size k if and only if G' contains a D-family-matching of weight $(D-2)Wm + 2(m+k)$.

$\Rightarrow$ Let $M \subseteq E$ be an induced matching of G of size at least k. W.l.o.g., assume that $M = \{e_1, \dots e_k\}$. Construct, for each $j \in \{1, \dots, k\}$, the cluster $S_j^M = \{v_a^1, v_b^1, e_j^1, \dots, e_j^{D-1}, v_a^2, v_b^2\}$, where a, b are such that $e_j = \{v_a, v_b\}$. Clearly S_j^M induces a graph of diameter D. Moreover, its weight is $(D-2)W + 4$. Now, for every edge $e_j \in E \setminus M$, let i_j be such that $v_{i_j} \in e_j$ and v_{i_j} does not belong to any edge from M, chosen arbitrarily if several choices are possible.

Notice that v_{i_j} is well defined, since M is an induced matching. Let $\mathcal{I} = \{i_j | e_j \in E \setminus M\}$. For every $i \in \mathcal{I}$, construct the cluster $S_i^O = \{v_i^1, v_i^2\} \cup \bigcup_{j|i_j=i} \{e_j^1, \dots, e_j^{D-1}\}$. Here again, one can check that the diameter of the graph induced by S_i^O is D, since it is either a path on $D+1$ vertices, or two or three paths of length $D+1$ whose respective endpoints have been identified. Finally, the sum of the weights of these clusters is $|E \setminus M|\,((D-2)W + 2)$. Hence, the total weight of this D-family-matching is $(D-2)Wm + 2(m+k)$, as desired.

$\Leftarrow$ Conversely, let $\mathcal{S}$ be a solution of weight at least $(D-2)Wm + 2(m+k)$. Because of the value of W, it holds that for every $j \in \{1, \dots, m\}$ and every $\ell \in \{1, \dots, D-2\}$, the edge $\{e_j^\ell, e_j^{\ell+1}\}$ belongs to some cluster of $\mathcal{S}$. Observe that the weight of these edges is $(D-2)Wm$, and the weight of all remaining edges, *i.e.* edges between V^1 and E^1, and edges between E^{D-1} and V^2, are 1. Hence, we may assume, w.l.o.g., that $\mathcal{S}$ contains at least $m+k$ edges among those between V^1 and E^1. Let M be the set of all $j \in \{1, \dots, m\}$ such that e_j^1 is in the same cluster as v_a^1 and v_b^1, where $e_j = \{v_a, v_b\}$. Observe that there cannot be $j, j' \in M$, $j \neq j'$, such that e_j^1 and $e_{j'}^1$ are in the same cluster, since the diameter of such a cluster would be at least $D+1$. Now, we show that we can modify the solution (without decreasing its weight) so that edges of index in M form an induced matching: suppose there exist $j, j' \in M$, $j \neq j'$, $r \notin M$ such that $e_r = \{v_a, v_b\}$ with $v_a \in e_j$ and $v_b \in e_{j'}$ (*i.e.* edges e_j and $e_{j'}$ are incident). We move v_a^1 from the cluster which contains e_j^1 to the cluster which contains e_r^1. Let us call $\mathcal{S}'$ the obtained solution. We have the following:

- $\mathcal{S}'$ is of same weight as $\mathcal{S}$;
- $\mathcal{S}'$ is a D-family-matching: firstly, v_a^1 is a vertex of degree 1 in the graph induced by its cluster in $\mathcal{S}$, so the diameter of the cluster containing e_j^1 is not greater in $\mathcal{S}'$. Secondly, we added a vertex of degree 1 in the cluster containing e_r^1. If this cluster has diameter at least $D+1$ in $\mathcal{S}'$, it implies that in $\mathcal{S}$, there exists $r' \neq r$ such that $e_{r'}^{D-1}$ and e_r^1 are in the same cluster, but since e_r^1 is only adjacent to e_r^2 in its cluster, the shortest path between e_r^1 and $e_{r'}^1$ is of length at least $2D-3$, a contradiction.

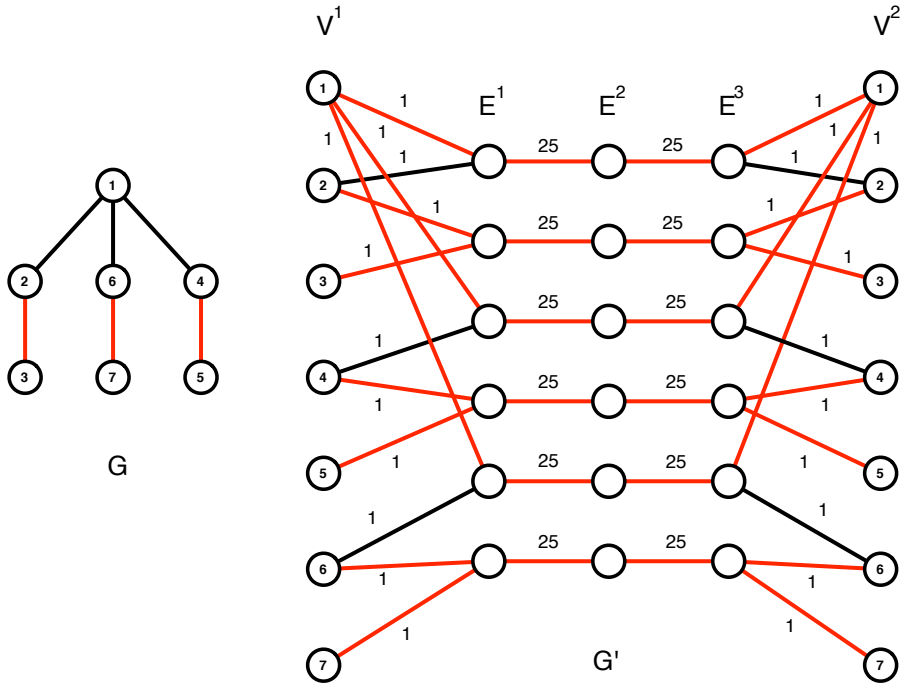
Applying this modification leads to a solution $\mathcal{S}$ such that M represents an induced matching in G. Finally, the number of edges contained in a cluster of $\mathcal{S}$ among those between V^1 and E^1 is at most $2|M| + |E \setminus M| = m + |M|$, and at least $m+k$, thus $|M| \geq k$. As previously, m can be bounded above by a linear function of the size of a maximum induced matching, which proves that the reduction is an L-reduction. □

To illustrate the proof of Theorem 8, consider the instance of Induced Matching problem depicted in Figure 6 (left). Figure 6 (right) represents the intersection graph constructed from it for $D = 4$. There is an induced matching of G of size 3 and there is a 2-family-matching composed of 4 sets with total weight equal to $18 + 12W = 318$.

3.2 Greedy strategies

Perhaps the most natural idea to solve the problem is a greedy approach, which would iteratively seek for maximal collections of subgraphs of diameter D. Unfortunately, we show that even for $D = 2$, there exist instances (having unary weights) for which picking a maximum biclique (complete bipartite graph) first leads to an arbitrarily bad solution.

Figure 6 Illustration of the proof of Theorem 8. See details in the text.



Lemma. 5. *For any integer $\lambda \geq 1$, there exists an intersection graph $G = (V, E, w)$ such that $\Phi_2(G)/\Phi(\mathcal{S}_{bic}) \geq \lambda - 2$, where $\mathcal{S}_{bic}$ is an optimal solution for the 2-family-matching problem among those containing a maximum biclique. Such a graph has unary weights, is of maximum degree λ, and contains $\lambda(\lambda-1)+1$ vertices.*

Proof of Lemma 5. Consider the intersection graph $G = (U, U', E, w)$ constructed as follows.

- Set $U = \{u_1, \ldots, u_\lambda\}$.
- Set $U' = \{u'_c\} \cup U'^1 \cup \ldots \cup U'^\lambda$, where $U'^i = \{u'^i_1, \ldots, u'^i_{\lambda-1}\}$ for every $i \in \{1, \ldots, \lambda\}$.
- Set $E = E^c \cup E^1 \cup \ldots \cup E^\lambda$, where
  - $E^c = \{\{u'_c, u_i\} \mid 1 \leq i \leq \lambda\}$
  - and $E^i = \{\{u_i, u'^i_j\} \mid 1 \leq j \leq \lambda - 1\}$ for every $i \in \{1, \ldots, n\}$.
- Set $w_e = 1$ for every $e \in E$.

Observe that the graph G is bipartite.

We now prove that the sub-graph of G with diameter at most 2 and that has the maximum number of edges is the graph $G[\{u'_c, u_i, \ldots, u_\lambda\}]$ composed of λ edges that is induced by the set of nodes $\{u'_c, u_i, \ldots, u_\lambda\}$. Indeed, suppose that node u'_c is not in such a graph. Then, if we remove u'_c from G, we obtain λ disjoint stars each composed of $\lambda - 1$ edges. Thus, since $w_e = 1$ for every $e \in E$, then we get that the graph $G[\{u'_c, u_i, \ldots, u_\lambda\}]$ induced by $\{u'_c, u_i, \ldots, u_\lambda\}$ maximizes the sum of the weights. Now, if we remove such a set from G, we get disjoint isolated nodes (that is each node has degree 0). We get that $\Phi(\mathcal{S}^{max}) = \lambda$.

We finally prove that there exists a 2-family-matching $\mathcal{S}$ for G such that $\Phi(\mathcal{S}) \geq \lambda(\lambda-2)$. Let $\mathcal{S} = \{S_1, \ldots, S_\lambda\}$ be such that $S_i = \{u_i\} \cup U'^i$ for every $i \in \{1, \ldots, \lambda\}$. Observe that $S_i \cap S_j = \emptyset$ for every $i, j \in \{1, \ldots, \lambda\}$, $i \neq j$. Furthermore, the graph $G[S_i]$ is a star and so has diameter 2. Thus, $\mathcal{S}$ is a $(\lambda, 2)$-family-matching for G. The number of edges of $G[S_i]$ is $|E(G[S_i])| = \lambda - 2$ for every $i \in \{1, \ldots, \lambda\}$. Since $w_e = 1$ for every $e \in E$, we finally get that

$$\Phi(\mathcal{S}^{max}) = \lambda, \quad \Phi(\mathcal{S}) \geq \lambda(\lambda-2), \text{and} \quad \frac{\Phi(\mathcal{S})}{\Phi(\mathcal{S}^{max})} = \lambda - 2.$$

This concludes the proof of Lemma 5. □

Notice that this result is in sharp contrast to the non-bipartite version of the problem, where we want to find a partition of the vertices of a graph in cliques which covers the maximum number of edges. It has been shown [5] that a greedy algorithm similar to the one described previously achieves a 2-approximation ratio if one is allowed to pick cliques of size at most some constant r only. Hence, this strategy gives a 2-approximation algorithm in graphs of maximum degree Δ, while it cannot achieve an approximation ratio better than Δ in the bipartite case. This gives the intuition that the complexity of the problem increases with the value of the diameter.

From this observation, another greedy strategy consists in first solving the problem with a smaller value of D. Here again, we show that such an algorithm cannot achieve a fixed approximation ratio. More precisely, we analyze the ratio between scores of optimal solutions for the D-family-matching problem for increasing values of the diameter, that is $\Phi_D(G)/\Phi_{D'}(G)$ for $D' < D$. Unfortunately, we show that this ratio is not bounded even for very simple classes of instances.

Lemma. 6. *For any integer $n \geq 1$, there exists an intersection graph $G = (V, E, w)$ composed of n nodes such that $\Phi_2(G)/\Phi_1(G) \geq n - 1$.*

Proof of Lemma 6. Let $t = n - 1$. Let $Z = \{z_1, \ldots, z_t\}$, $F = \{F_1\}$, and $F' = \{F'_1, \ldots, F'_t\}$ be an instance of the D-family-matching problem, where $F_1 = \{z_1, \ldots, z_t\}$ and $F'_i = \{z_i\}$ for every $i \in \{1, \ldots, t\}$. The intersection graph $G = (U, U', E, w)$ is such that $U = \{u_1\}$, $U' = \{u'_1, \ldots, u'_t\}$, $E = \{\{u_1, u'_i\} \mid 1 \leq i \leq t\}$, and $w_e = 1$ for every $e \in E$. We first prove that any solution $\mathcal{S}^{D=1}$ for the 1-family-matching problem is such that $\Phi(\mathcal{S}^{D=1}) \leq 1$. Indeed, every sub-graph of G with diameter at most 1 is composed of at most 1 edge. Furthermore, if $\mathcal{S}^{D=1}$ contains a set S that induces a sub-graph composed of one edge, then all others sets induce sub-graphs that do not contain any edge (observe that every edge contains the central node u_1 of the star graph G). Otherwise the family $\mathcal{S}^{D=1}$ does not satisfy the property that the subsets are disjoint. Thus, $\Phi(\mathcal{S}^{D=1}) \leq 1$. Finally, let $\mathcal{S}^{D=2} = \{U \cup U'\}$. The graph induced by $U \cup U'$ is the graph G that has diameter 2 and $\Phi(\mathcal{S}^{D=2}) = t = n - 1$. □

In the next sections, we provide polynomial-time algorithms for simple graphs classes, and then use some of them to obtain efficient algorithms in general bipartite graphs.

4 Polynomial time dynamic programming algorithms for some classes

In this section, we prove polynomial-time complexity exact dynamic programming algorithms for the D-family-matching problem for some classes of graphs: trees, paths, cycles, graphs of maximum degree two.

4.1 The D-family-matching problem for trees

We explain our exact polynomial time dynamic programming algorithm to solve the D-family-matching problem when the graph is a tree.

Theorem. 9 (Computation of $\Phi_D(G)$ for trees)**.** *Let $D \in \mathbb{N}^+$. Consider any intersection tree $T = (V, E, w)$ of maximum degree $\Delta \geq 0$. Then, there exists an $O(D^2\Delta^2 n)$-time complexity algorithm for the D-family-matching problem for T.*

Proof. Consider the tree T rooted at any node $r \in V$. We call this rooted tree T_r. Given any node $v \in V$, let T_v be the sub-tree of T_r rooted at v such that $V(T_v)$ contains all the nodes $v' \in V$ such that there is a simple path between v' and r in T_r that contains v in T_r. A simple path is a path such that each node is contained at most once in it. We define the function Ψ_D as follows. For every $v \in V$ and every $i \in \{-1, 0, \ldots, D\}$, then $\Psi_D(T_v, i)$ is the score of an optimal solution $\mathcal{S}$ for the D-family-matching problem, for the intersection tree T_v, such that:

- if $i \geq 0$, then there exists $S \in \mathcal{S}$, $v \in S$, and the sub-tree induced by the set of nodes S has depth at most i;
- if $i = -1$, then for every $S \in \mathcal{S}$, we have $v \notin S$.

Note that $\Psi_D(T_v, 0)$ is the score of an optimal solution $\mathcal{S}$ when $\{v\} \in \mathcal{S}$ (say otherwise, v is alone in a set). In the following, we abuse the notation writing $\Psi_D(v, i)$ instead of $\Psi_D(T_v, i)$.

First of all, for every leaf $v \in V$ of T_r and every $i \in \{-1, 0, \ldots, D\}$, then $\Psi_D(v, i) = 0$. A leaf is a node of degree one and different than the root r.

Let $v \in V$ be any node that is not a leaf. Let $N(v) = \{v_1, \ldots, v_q\}$ be the set of $q \geq 1$ neighbors of v in T_v. Suppose we have computed $\Psi_D(v_j, i)$ for every $j \in \{1, \ldots, q\}$ and every $i \in \{-1, \ldots, D\}$. We prove that we can compute $\Psi_D(v, i)$ for every $i \in \{-1, \ldots, D\}$. The computation is divided into two different cases (claims).

Claim 10. *For every $i \in \{-1, 0\}$, then*

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

Proof of Claim 10. Let us first consider $i = -1$. We consider here an optimal solution for the D-family-matching problem for T_v such that v does not belong to any set. Thus, we compute for every sub-tree T_{v_j}, $1 \leq j \leq q$, the score of an optimal solution for the D-family-matching problem for T_{v_j}. Note that the depth of the sub-tree (set) rooted at v_j in the solution has no importance here. The score of such a score is $\max_{i' \in \{0,\ldots,D\}} \Psi_D(v_j, i')$. Then, $\Psi_D(v, -1)$ is the sum of all such scores.

Let us now consider $i = 0$. It means that we consider an optimal solution for the D-family-matching problem for T_v such that v is alone in a set. Observe that $\Psi_D(v, 0) = \Psi_D(v, -1)$. □

Claim 11. *For every $i \in \{1, \ldots, D\}$, then*

$$\Psi_D(v, i) = \max_{j \in \{1,\ldots,q\}} (\Psi_D(v_j, i-1) + w_{v,v_j} +$$

$$\sum_{j' \in \{1,\ldots,q\} \setminus \{j\}} \max(\max_{i' \in \{1,\ldots,\min(i-1, D-i-1)\}} \Psi_D(v_{j'}, i') + w_{v,v_{j'}}, \max_{i' \in \{1,\ldots,D\}} \Psi_D(v_{j'}, i'))).$$

Proof of Claim 11. We compute here the score $\Psi_D(v,i)$ of an optimal solution for the D-family-matching problem for T_v such that the depth of the sub-tree (set) that contains v in the solution is exactly i. We denote by S_v the set of nodes of such a sub-tree. To do that, we first need to choose one sub-tree T_{v_j}, for some $j \in \{1,\dots,q\}$, such that the set (sub-tree) that contains v_j in the solution for T_v, is such that the sub-tree induced by $S_v \cap V(T_{v_j})$ has depth $i-1$. In order to compute such j, we enumerate the q different possibilities. For every possible choice ($j = 1,\dots,q$), we compute the largest possible score. Such a score is $\Psi_D(v_j, i-1)$ plus the the weight w_{v,v_j} of the edge $\{v,v_j\}$ plus the largest possible score for the other neighbors of v. More precisely, for every $j' \in \{1,\dots,q\}$, $j' \neq j$, there are two cases.

- $S_v \cap V(T_{v_j}) = \emptyset$. In that case, the largest possible score corresponding to the sub-tree $T_{v_{j'}}$ is $\max_{i' \in \{1,\dots,D\}} \Psi_D(v_{j'}, i')$.
- $S_v \cap V(T_{v_j}) \neq \emptyset$. In that case, the largest possible score is $\max_{i' \in \{1,\dots,\min(i-1,D-i-1)\}} \Psi_D(v_{j'}, i') + w_{v,v_{j'}}$. Indeed, we add the weight $w_{v,v_{j'}}$ by assumption and we then compute the score of an optimal solution for the D-family-matching problem for $T_{v_{j'}}$ such that $v_{j'}$ is in a sub-tree (set) of depth at most $D-i-1$ and also at most $i-1$. Otherwise, the diameter of S_v would be at least $D+1$ and/or the depth would be at least $i-1$.

We determine the maximum score between these two scores. We finally obtain an optimal score and we determine a best choice for j in order to compute $\Psi_D(v,i)$. □

For every $v \in V$, the time complexity of the computation of $\Psi_D(v,i)$, for all $i \in \{-1,\dots,D\}$, is $O(qD)$ for the first case and $O(q^2D^2)$ for the second case. We get that the time complexity of the algorithm is $O(D^2\Delta^2 n)$. Note that $\Delta \leq n-1$ and $D \leq n-1$. Finally, when we have computed $\Psi_D(r,i)$ for every $i \in \{-1,\dots,D\}$, we can deduce an optimal solution $\mathcal{S}$ for the D-family-matching problem for T. Indeed, $\Phi(\mathcal{S}) = \Phi_D(G) = \max_{i \in \{-1,\dots,D\}} \Psi_D(r,i)$. □

Remark 1. *The algorithm from Theorem 9 computes a solution for a tree. This solution can be enriched at no cost by adding edges whose endpoints belong to the same meta-cluster. In general, the intersection graph is indeed not a tree, so that such edges are unaccounted for.*

4.2 The D-family-matching problem for paths

We illustrate the result of Theorem 9 by considering paths. Consider an intersection path $G = (V,E,w)$. By Theorem 9, given $D \geq 1$, there is an $O(D^2 n)$-time complexity algorithm for the D-family-matching problem because $\Delta = 2$. We prove in Lemma 7 a better time complexity algorithm for the D-family-matching problem. Indeed, the time complexity is $O(Dn)$.

Lemma. 7 (Computation of $\Phi_D(G)$ for paths)**.** *Let $D \in \mathbb{N}^+$. Consider any intersection path $G = (V,E,w)$. Then, there exists an $O(Dn)$-time complexity algorithm for the D-family-matching problem for G.*

Proof of Lemma 7. Let $V = \{v_1,\dots,v_n\}$. Let $E = \{\{v_j, v_{j+1}\} \mid 1 \leq j \leq n-1\}$. We define the function Ψ_D as follows. For every $t \in \{1,\dots,n\}$ and every $i \in \{\max(1,t-D),\dots,t+1\}$, then $\Psi_D(v_t,i)$ is the score of an optimal solution $\mathcal{S}$ of the D-family-matching problem, for the sub-path induced by the set of nodes $\{v_1,\dots,v_t\}$, such that $\{v_i,\dots,v_t\} \in \mathcal{S}$. The case $i = t+1$ means that v_t does not belong to any set. Note that we consider $i \geq \max(1,t-D)$ because, otherwise we would not have an admissible solution (because of the diameter constraint). First of all, $\Psi_D(v_1,1) = \Psi_D(v_1,2) = 0$.

Let $t \in \{1,\dots,n-1\}$. Suppose we have computed $\Psi_D(v_{t'},i)$ for every $t' \in \{1,\dots,t\}$ and every $i \in \{\max(1,t-D),\dots,t'+1\}$. We prove that we can compute $\Psi_D(v_{t'+1},i)$ for every

$i \in \{\max(1, t-D), \ldots, t'+1\}$ in $O(D)$-time. There are two different cases (corresponding to the two following claims).

Claim 12. *For every $i \in \{\max(1, t+1-D), \ldots, t\}$, then*

$$\Psi_D(v_{t+1}, i) = w_{v_t, v_{t+1}} + \Psi_D(v_t, i).$$

Proof of Claim 12. The set of nodes $\{v_i, \ldots, v_{t+1}\}$, $\max(1, t+1-D) \leq i \leq t$, must be a set of the solution. Thus, we have to consider the optimal solution for the D-family-matching problem, for the sub-path induced by the set of nodes $\{v_i, \ldots, v_t\}$, such that $\{v_i, \ldots, v_t\}$ is a set of this solution. We then modify this solution by adding node v_{t+1} in the last set, and we obtain the optimal solution for the D-family-matching problem, for the sub-path induced by the set of nodes $\{v_i, \ldots, v_t\}$, such that $\{v_1, \ldots, v_{t+1}\}$ is a set of this solution. □

Claim 13.

$$\Psi_D(v_{t+1}, t+1) = \Psi_D(v_{t+1}, t+2) = \max_{i \in \{\max(1, t-D), \ldots, t+1\}} \Psi_D(v_t, i).$$

Proof of Claim 13. We first prove the result for $\Psi_D(v_{t+1}, t+1)$. Any solution must contain the set $\{v_{t+1}\}$. Thus, we have to consider an optimal solution for the D-family-matching problem for the sub-path induced by the set of nodes $\{v_i, \ldots, v_t\}$.

We now prove the result for $\Psi_D(v_{t+1}, t+2)$. Since node $\{v_{t+1}\}$ does not belong to any set, then we have to consider again an optimal solution for the D-family-matching problem for the sub-path induced by the set of nodes $\{v_i, \ldots, v_t\}$. □

For every $t \in \{1, \ldots, n\}$, we address the time complexity of computing Ψ as follows. For each claim, the time complexity of the computation of Ψ is $O(D)$. We get that the time complexity of the dynamic programming algorithm is $O(nD)$.

To conclude the proof of Lemma 7, when we have computed $\Psi_D(v_n, i)$ for every $i \in \{\max(1, n-D), \ldots, n+1\}$, then we can deduce an optimal solution $\mathcal{S}$ and the optimal value for the D-family-matching problem for G. Indeed,

$$\Phi_D(G) = \max_{i \in \{\max(1, n-D), \ldots, n+1\}} \Psi_D(v_n, i).$$

Recall that $n+1$ means that node v_n does not belong to any set of the solution. □

4.3 The D-family-matching problem for cycles

We now deduce in Corollary 1 an efficient algorithm for the D-family-matching problem when G is an even cycle.

Corollary. 1 (Computation of $\Phi_D(G)$ for cycles)**.** *Let $D \in \mathbb{N}^+$. Consider any intersection graph $G = (V, E, w)$ that is an even cycle. Then, there exists an $O(D^2 n)$-time complexity algorithm for the D-family-matching problem for G.*

Indeed, we have

$$\Phi_D(G) = \max_{H \in H(G, v)} (\Psi_D(G_H) + \sum_{e \in E_H} w_e).$$

We address in Corollary 2 the results in terms of the equivalent definition proved in Section 2.1.

Corollary. 2. *Let $D \in \mathbb{N}^+$. Consider any instance of the D-family-matching problem such that:*

- *for every* $i \in \{1,\dots,r\}$*, there exist* $j_1, j_2 \in \{1,\dots,r'\}$ *such that* $F_i \cap F'_j = \emptyset$ *for any* $j \in \{1,\dots,r'\} \setminus \{j_1, j_2\}$.
- *for every* $j \in 1,\dots,r'\}$*, there exist* $i_1, i_2 \in \{1,\dots,r\}$ *such that* $F'_j \cap F_i = \emptyset$ *for any* $i \in \{1,\dots,r\} \setminus \{i_1, i_2\}$.

Then, there exists an $O((r+r')D^2)$*-time complexity algorithm for the* D*-family-matching problem.*

Say otherwise, Corollary 2 shows that there is a polynomial time algorithm for the D-family-matching problem if any set in $F \cup F'$ has a non-empty intersection with at most two other sets of $F \cup F'$.

Notice finally that the problem can be solved in $O(|cc(G)| \max_{C \in cc(G)} f(C))$ for G if D-family-matching can be solved in $O(f(C))$ time for any $C \in cc(G)$, where $cc(G)$ denotes the set of maximal connected components of G.

5 Generic approach based on spanning trees

In this section, we provide a generic approach for solving the problem in general instances. This approach relies on computing a solution having a particular structure defined by a given spanning tree T of the input graph. Formally:

Definition. 8 (D-family-matching constrained by a tree)**.** *Let* $D \in \mathbb{N}^+$*. Let* $G = (V, E, w)$ *be an intersection graph and* T *be a spanning tree of* G*. A* D*-family-matching for* G *constrained by* T *is a* D*-family-matching* $\mathcal{S}$ *for* G *such that all* $S \in \mathcal{S}$ *induces a connected subtree in* T*.*

We thus obtain the following sub-problem of D-family-matching.

Definition. 9 (D-family-matching problem constrained by a tree)**.** *Let* $D \in \mathbb{N}^+$*. Given an intersection graph* G *and a spanning tree* T *of* G*, the* D*-family-matching problem consists in computing* $\Phi_D(G,T) = \max_{\mathcal{S} \in \mathcal{S}_D(G,T)} \Phi(\mathcal{S})$*, where* $\mathcal{S}_D(G,T)$ *is the set of all* D*-family-matching constrained by* T*.*

We are now ready to define our generic algorithm (Algorithm 1). Informally, it iteratively generates a spanning tree T of G, and compute a D-family-matching constrained by T. Let us describe the main ingredients of Algorithm 1 by explaining the three parameters needed.

- **A property** $\Pi(\mathcal{M})$, depending on the set $\mathcal{M}$ of already computed D-family-matchings, represents the halting condition of the algorithm.
- **A spanning tree generator** $\mathcal{R}(G, \lambda)$ computes the rooted spanning tree T^λ of G that is used at step $\lambda \geq 1$ by Algorithm $\mathcal{A}$.
- **An algorithm** $\mathcal{A}(G, T^\lambda, D)$ computes a D-family-matching $\mathcal{S}^\lambda$ constrained by T^λ.

The interest of this approach is twofold. The first one is the fact that solving optimally the D-family-matching constraint by T, for every spanning tree T, leads to an optimal solution of the general D-family-matching problem. This is the point of the following result.

Lemma. 8. *Let* $D \in \mathbb{N}^+$*. Let* G *be any intersection graph. Then, there exists a rooted spanning tree* T *of* G *such that* $\Phi_D(G) = \Phi_D(G,T)$*.*

Proof of Lemma 8. For some $k \geq 1$, consider an optimal solution $\mathcal{S} = \{S_1, \dots, S_k\}$ for the D-family-matching problem for G. For every $i \in \{1,\dots,k\}$, let T_i be any spanning tree of $G[S_i]$. Let T be any rooted spanning tree of G such that $E(T_i) \subseteq E(T)$ for every $i \in \{1,\dots,k\}$. By construction of T, $\mathcal{S}$ is an admissible solution for the D-family-matching problem for G constrained by T. Thus, $\Phi_D(G,T) = \Phi_D(G)$. □

Algorithm 1 Generic algorithm for the D-family-matching problem.

Require: An intersection graph $G = (V, E, w)$, an integer $D \geq 1$, a property Π, a spanning tree generator $\mathcal{R}$, and an algorithm $\mathcal{A}$.

1: $\mathcal{M} := \emptyset$, $\lambda := 0$
2: **while** $\neg\ \Pi(\mathcal{M})$ **do**
3: $\quad\lambda := \lambda + 1$; Compute the spanning tree $T^\lambda := \mathcal{R}(G, \lambda)$
4: $\quad$Compute $\mathcal{S}^\lambda$ by using Algorithm $\mathcal{A}(G, T^\lambda, D)$; $\mathcal{M} := \mathcal{M} \cup \mathcal{S}^\lambda$
5: **return** $\mathcal{S} \in \mathcal{M}$ of maximum score

Then, we show that it is possible, given a spanning tree T, to compute an optimal D-family-matching constrained by T in an efficient way, provided the diameter D and the maximum degree Δ of the input graph are bounded by a constant.

Lemma. 9 (Computation of $\Phi_D(G, T)$)**.** *Let $D \in \mathbb{N}^+$. Let $G = (V, E, w)$ be any intersection graph and T be any spanning tree of G. Then, there exists a $O(2^{D\Delta \log_2(\Delta)}n)$-time algorithm for the D-family-matching problem for G constrained by T.*

Proof of Lemma 9. Consider the tree T rooted at any node $r \in V$ such that r has not degree Δ. Such a node always exist if T contains at least three nodes. We call this rooted tree T_r. We define the function Ψ_D as follows. For every $v \in V$ and every $H \in H(G, T_r, v)$, then $\Psi_D(v, H)$ is the score of an optimal solution $\mathcal{S}$ for the D-family-matching problem, for the graph $G[V(t_v)]$ induced by the set of nodes $V(T_v)$, constrained by T_v, and such that $V(H) \in \mathcal{S}$. We allow H to be the empty graph $(\emptyset, \emptyset)$.

By convention, if there is no admissible solution, we set $\Psi_D(v, H) = -\infty$.

First of all, for every leaf $v \in V$ of T_r, then

- $\Psi_D(v, H) = 0$ if $H \in \{(\emptyset, \emptyset), (\{v\}, \emptyset)\}$,
- $\Psi_D(v, H) = -\infty$ if $H \in \{(\emptyset, \emptyset), (\{v\}, \emptyset)\}$.

A leaf is a node of degree one and different than the root r.

Let $v \in V$ be any node that is not a leaf. Let $N(v) = \{v_1, \ldots, v_q\}$ be the set of $q \geq 1$ neighbors of v in T_v. Suppose we have computed $\Psi_D(v_j, H)$ for every $j \in \{1, \ldots, q\}$ and every $H \in H(G, T_r, v_j)$. We prove that we can compute $\Psi_D(v, H)$ for every $H \in H(G, T_r, v)$. There are three different cases (corresponding to the three following claims).

Claim 14.

$$\Psi_D(v, (\emptyset, \emptyset)) = \max_{(H_1, \ldots, H_q) \in \mathcal{H}} \sum_{j=1}^{q} \Psi_D(v_j, H_j),$$

where $\mathcal{H}$ is the set of all vectors $(H_1, \ldots, H_q)$ such that $H_i \in H(G, T_r, v_i)$ for every $i \in \{1, \ldots, q\}$.

Proof of Claim 14. We consider here an optimal solution for the D-family-matching problem for the graph $G[V(t_v)]$ induced by the set of nodes $V(T_v)$, constrained by T_v, and such that v does not belong to any set. Thus, $\Psi_D(v, (\emptyset, \emptyset))$ consists in choosing the set H_j that contains v_j (possibly empty) for every $j \in \{1, \ldots, q\}$, such that the score is maximal. □

Claim 15.

$$\Psi_D(v, (\{v\}, \emptyset)) = \max_{(H_1, \ldots, H_q) \in \mathcal{H}} \sum_{j=1}^{q} \Psi_D(v_j, H_j),$$

where $\mathcal{H}$ is the set of all vectors $(H_1, \ldots, H_q)$ such that $H_i \in H(G, T_r, v_i)$ for every $i \in \{1, \ldots, q\}$.

Proof of Claim 15. This proof is similar to the proof of Claim 14. Indeed, we consider an optimal solution $\mathcal{S}$ for the D-family-matching problem for the sub-graph induced by the set of nodes $V(T_v)$, constrained by T_v, and such that $\{v\} \in \mathcal{S}$. Thus, $\Psi_D(v, (\{v\}, \emptyset))$ consists in choosing the set H_j that contains v_j (possibly empty) for every $j \in \{1, \ldots, q\}$, such that the score is maximal. □

Claim 16. *Let $H \in H(G, T_r, v)$ be any sub-tree. Without loss of generality, assume that, for some q', $V(H) \cap V(T_{v_j}) \neq \emptyset$ for every $j \in \{1, \ldots, q'\}$, and $V(H) \cap V(T_{v_j}) = \emptyset$ for every $j \in \{q'+1, \ldots, q\}$. Let H_{v_j} be the intersection between H and the sub-tree T_{v_j}, that is $V(H_{v_j}) = V(H) \cap V(T_{v_j})$. Then*

$$\Psi_D(v, H) =$$

$$\sum_{e' \in E(H'_v)} w_{e'} \; + \sum_{j=1}^{q'} (\Psi_D(v_j, H_{v_j}) - \sum_{e' \in E(H_{v_j})} w_{e'}) \; + \max_{(H_{q'+1}, \ldots, H_q) \in \mathcal{H}'} \sum_{j=q'+1}^{q} \Psi_D(v_j, H_j),$$

where $\mathcal{H}$ is the set of all vectors $(H_{q'+1}, \ldots, H_q)$ such that $H_i \in H(G, T_r, v_i)$ for every $i \in \{q'+1, \ldots, q\}$.

Proof of Claim 16. We consider an optimal solution $\mathcal{S}$ for the D-family-matching problem for the graph $G[V(t_v)]$ induced by the set of nodes $V(T_v)$, constrained by T_v, and such that $V(H) \in \mathcal{S}$. The sub-tree H contains v and q' sub-trees $H^{v_1}, \ldots, H^{v_{q'}}$ rooted at $v_1, \ldots, v_{q'}$, respectively. Thus, $\Psi_D(v, H)$ consists in choosing, for every $j \in \{q'+1, \ldots, q\}$, the set H_j that contains v_j (possibly empty) in order to maximize the value of the solution. Note that H_j must be a graph that belongs to $H(G, T_r, v_j)$ by definition of the problem constrained by a tree. □

For every $v \in V$, we address the time complexity of computing Ψ as follows. The time complexity of the computation done in Claim 14 is $O(\prod_{j=1}^{q} h(G, T_r, v_i))$. The time complexity of the computation done in Claim 15 is $O(\prod_{j=1}^{q} h(G, T_r, v_i))$. The time complexity of the computation done in Claim 16 is $O(h(G, T_r, v)(q' + |E(H_{v_j})| + \prod_{j=q'+1}^{q} h(G, T_r, v_i)))$. Since $h(G, T_r, v) \leq h(G, T_r)$ for every $v \in V$ and $q \leq \Delta - 1$ by the choice of the root of T, then we get that the time complexity of the algorithm is $O(h(G, T_r)^{\Delta} n) = O(2^{D\Delta \log_2(\Delta)} n)$ because $h(G, T) = O(2^{\Delta^D})$.

To conclude the proof of Lemma 9, when we have computed $\Psi_D(r, H)$ for every $H \in H(G, T_r, r)$, we can deduce an optimal solution $\mathcal{S}$ and the optimal value of the D-family-matching problem for G constrained by T. Indeed,

$$\Phi_D(G, T_r) = \max_{H \in H(G, T_r, r)} \Psi_D(r, H).$$

Note that H can be empty (in that case r does not belong to any set of $\mathcal{S}$). □

Let us first introduce some notations. For every $v \in V$, let $H(G, v)$ be the set of all different sub-graphs of G that contain v and of diameter at most D. Let $H(G) = \cup_{v \in V} H(G, v)$. We define $h(G, v) = |H(G, v)|$ for every $v \in V$ and $h(G) = \max_{v \in V} h(G, v)$. Let T_r be any spanning tree of G rooted at node $r \in V$. For every $v \in V$, we define $H(G, T_r, v)$ as the set of all $H \in H(G, v)$ such that the graph induced by the set of nodes $V(H) \cap V(T_v)$ is a (connected) sub-tree rooted at v. Let $H(G, T_r) = \cup_{v \in V} H(G, T_r, v)$. We define $h(G, T_r, v) = |H(G, T_r, v)|$ for every $v \in V$ and $h(G, T_r) = \max_{v \in V} h(G, T_r, v)$. Furthermore, let $\mathcal{T}(G)$ be the set of all different rooted spanning trees of G.

Corollary. 3. *Given any positive integer $D \geq 1$ and any intersection graph G, Algorithm 1 returns $\Phi_D(G)$, that is an optimal solution for the D-family-matching problem for G, if:*

- $\Pi(\mathcal{M}) \Leftrightarrow |\mathcal{M}| = |\mathcal{T}(G)|$,
- $\mathcal{R}(G, \lambda) = T^\lambda$, *where* $\mathcal{T}(G) = \{T^1, \ldots, T^{|\mathcal{T}(G)|}\}$,
- *and Algorithm* $\mathcal{A}(G, T^\lambda, D)$ *returns* $\Phi_D(G, T^\lambda)$ *(Lemma 9).*

Furthermore, the time complexity of Algorithm 1 is $O(|\mathcal{T}(G)| \max_{T_r \in \mathcal{T}(G)} h(G, T_r)^\Delta n)$.

Lemma. 10. *Let G be any intersection graph. Then, there exists a rooted spanning tree T of G such that $\Phi_2(G) \leq 2\Delta\Phi_2(T)$.*

Proof of Lemma 10. For some $k \geq 1$, consider an optimal solution $\mathcal{S} = \{S_1, \ldots, S_k\}$ for the 2-family-matching problem for G. For every $i \in \{1, \ldots, k\}$, let T_i be a maximum spanning tree of $G[S_i]$. Let T be any rooted spanning tree of G such that $E(T_i) \subseteq E(T)$ for every $i \in \{1, \ldots, k\}$. For every $i \in \{1, \ldots, k\}$, we have $\Delta \sum_{e \in E(T_i)} w_e \geq \sum_{e \in E(G[S_i])} w_e$. Indeed, since $D = 2$, $G[S_i]$ is necessarily a complete bipartite graph and its number of nodes is at most 2Δ. It is sufficient to select the maximum star as T_i in order to get the inequality. Thus, by construction of T, the algorithm returns at least the desired score. □

Corollary. 4. *Given any intersection graph G, Algorithm 1 returns a 2Δ-approximation for the 2-family-matching problem for G if:*

- $\Pi(\mathcal{M}) \Leftrightarrow |\mathcal{M}| = |\mathcal{T}(G)|$,
- $\mathcal{R}(G, \lambda) = T^\lambda$, *where* $\mathcal{T}(G) = \{T^1, \ldots, T^{|\mathcal{T}(G)|}\}$,
- *and Algorithm* $\mathcal{A}(G, T^\lambda, D)$ *returns* $\Phi_D(T^\lambda)$ *(Theorem 9).*

Furthermore, the time complexity of Algorithm 1 is $O(|\mathcal{T}(G)| D^2 \Delta^2 n)$.

Finally, we obtain an efficient (polynomial) heuristic using our dynamic programming algorithm for trees described in Theorem 9. Indeed, an optimal solution for the D-family-matching problem on a spanning tree T is a D-family-matching for G constrained by T. We present an implementation of this heuristic and results of experiments in the next section.

6 On the choice of D

6.1 Rationale

As motivated in section 1.3, parameter D acts as a scale parameter providing information of the structure of the intersection graph and on the scale at which clusters coalesce. When the intersection graph is dense or has a specific topology (star-shaped), trivial values of Φ are obtained for small values of D, and a unit change of D may trigger an abrupt change of Φ. However, in more complex situations, large values of D may be required.

As a general strategy to choose D, we suggest identifying drops in Φ when changing D. Indeed, a solution is to find range(s) of D corresponding to a *plateau* for Φ, in order to get subset of diameters that give similar Φ. Phrased differently, consider a set of pairs $\{(D, \Phi_D)\}$ for consecutive values of D. In the 2D space (D, Φ_D), consider the bounding box of these points: the longer and the thinner this bounding box, the better. In the sequel, we present three strategies to compute such plateaus:

- Section 6.2: a strategy computing a set of non-overlapping plateaus optimizing a functional favoring *long* and *thin* plateaus.

- Section 6.3: a strategy allowing the user to specify the number of plateaus to be obtained.
- Section 6.4: a strategy delivering a hierarchical decomposition into plateaus, which is of interest if there are several *vertical scales*.

We note in passing that optimizing the width and/or height of plateaus is similar in spirit to other strategies which were developed in particular to estimate the *correct* number of clusters: the elbow method tracks jumps of the derivative of the objective function [28]; the gap method uses the change in within-cluster dispersion [42]; the split-and-merge procedure for `k-means` [1] requires a parameterization of high-density areas; the integrated completed likelihood (ICL) method [3] tracks significant drops in the entropic penalty complementing the BIC criterion; the density based - mode seeking method exploits a separation gap (in the persistence diagram) for local maxima [6], etc.

6.2 Computation of tradeoff-plateaus of large widths and small heights

We first develop a strategy to compute plateaus of large width and small height. More precisely, we develop a quadratic time algorithm for the problem of computing the diameter D that minimizes a tradeoff between the widths and the heights of the plateaus.

Definition. 10 (Tradeoff-plateau problem)**.** *Let τ_w and τ_h be two increasing functions of one variable such that $\tau_w(y) > 0$ and $\tau_h(y) > 0$ for any y. Given a set $\{\Phi_1(G), \ldots, \Phi_{D_G}(G)\}$, the Tradeoff-plateau problem consists in determining $\mu \in \{1, \ldots, D_G\}$ plateaus (intervals) $I_1, \ldots, I_\mu$ of $[1, D_G]$ such that $I_1 \cup \ldots \cup I_\mu = \{1, \ldots, D_G\}$, $I_x \cap I_{x'} = \emptyset$ for every $1 \leq x < x' \leq \mu$, and such that the following function is minimum:*

$$-\sum_{x=1}^{\mu} \frac{\tau_w(|I_x|)}{\tau_h(\max_{D,D' \in I_x \cap \mathbb{N}} \Phi_{D'}(G) - \Phi_D(G))},$$

where $|I_x|$ is the size of plateau I_x.

Note that as the simplest choices for τ_w and τ_h, one can use two simple linear functions of the width and height of the bounding box containing the consecutive points $\{(D, \Phi_D)\}$ (recall that we must have $\tau_h(y) > 0$). This yields the following objective function

$$-\sum_{x=1}^{\mu} \frac{|I_x| - 1}{\max_{D,D' \in I_x \cap \mathbb{N}} \Phi_{D'}(G) - \Phi_{D+1}(G) + 1}.$$

The following theorem proves in a constructive way, using dynamic programming, that the computation of such plateaus can be done in quadratic time:

Theorem. 17. *There is an $O(D_G^2)$-time complexity algorithm that computes an optimal solution for the Tradeoff-plateau problem.*

Proof of Theorem 17. We first pre-compute $\frac{\tau_w(|I_x|)}{\tau_h(\max_{D,D' \in I_x \cap \mathbb{N}} \Phi_{D'}(G) - \Phi_D(G))}$ for every possible plateau I_x. There are $O(D_G^2)$ such computations. Then, we prove a direct dynamic programming algorithm. For every $D \in \{1, \ldots, D_G\}$, $\rho(D)$ is the score of an optimal solution for the tradeoff-plateau problem for the set $\{\Phi_1(G), \ldots, \Phi_D(G)\}$. We have $\rho(0) = 0$. In general, for every $D \in \{1, \ldots, D_G - 1\}$, we have

$$\rho(D+1) = \min_{i \in \{0,\ldots,D\}} (\rho(i) - \frac{\tau_w(D-i)}{\tau_h(\max_{i,i' \in [i+1,D+1]} \Phi_{i'}(G) - \Phi_i(G))}).$$

Recall that τ_w and τ_h are two increasing functions of one variable such that $\tau_w(y) > 0$ and $\tau_h(y) > 0$ for any y. If $|I_x| = 1$, then $\frac{\tau_w(D-i)}{\tau_h(\max_{i,i' \in I_x \cap \mathbb{N}} \Phi_{i'}(G) - \Phi_i(G))} = 0$. The computation consists of computing an optimal solution for every possible plateau $[i+1, D+1]$ and, among these optimal solutions, calculate an optimal one of score $\rho(D+1)$. For every $D \in \{1, \ldots, D_G - 1\}$, there are $O(D)$ optimal solutions to compute, and each of them can be computed in constant time. Finally, $\rho(D_g)$ is the optimal score and the total time computation is quadratic in the diameter D_G. □

Once plateaus have been computed, the *plateau plot* displays one rectangle (containing the corresponding points (D, Φ_D)) for each plateau. Since gaps between plateaus are of interest, we define:

Definition. 11. *(Increment in a plateau plot) The increment of a plateau is defined as the mean value of its scores, minus the average value computed on the preceding plateau.*

We note that a natural choice for the value of D is the last plateau with a *significant* increment – in a spirit analogous to the strategies recalled in section 6.1.

Remark 2. *In practice, the range of D values explored is set to $1, \ldots, D_G$, with D_G the diameter of the intersection graph. When $D = D_G$ an exact algorithm solving the D-family matching problem would return a number of meta-clusters corresponding to the number of connected components of the intersection graph – the exact solution. However, since we only use a heuristic based on spanning trees, at $D = D_G$, the number of meta-clusters may be over-estimated. However this is not an issue since for $D = D_G$, the exact solution is known.*

6.3 Computation of multiple sets of plateaus of small heights

We develop an algorithm computing an optimal set of μ plateaus in terms of sum of heights (of the plateaus) for every possible value for $\mu \in \{1, \ldots, D_G\}$.

Definition. 12 (Plateau problem)**.** *Let $G = (V, E)$ be a graph of diameter D_G. Given a set $\{\Phi_1(G), \ldots, \Phi_{D_G}(G)\}$ and a number of plateaus $\mu \in \{1, \ldots, D_G\}$, the Plateau problem consists in determining μ plateaus (intervals) $I_1, \ldots, I_\mu$ of $[1, D_G]$ such that $I_1 \cup \ldots \cup I_\mu = \{1, \ldots, D_G\}$, $I_x \cap I_{x'} = \emptyset$ for every $1 \leq x < x' \leq \mu$, and such that the following function is minimum:*

$$\sum_{x=1}^{\mu} \max_{D, D' \in I_x \cap \mathbb{N}} \Phi_{D'}(G) - \Phi_D(G).$$

Theorem. 18. *There is an $O(D_G^5)$-time complexity algorithm that computes an optimal solution for the Plateau problem for every $\mu \in \{1, \ldots, n\}$.*

Proof of Lemma 11. We prove the result by induction. Clearly, $\rho_{1,\Phi_1(G),\Phi_1(G)}(1) = 0$.

Assume that we have computed $\rho_{y,x^-,x^+}(D)$ for every $D \in \{1, \ldots, D'\}$, for every $y \in \{1, \ldots, D\}$, for every $x^-, x^+ \in \mathcal{P}_D$ with $x^- \leq x^+$. We prove that the five cases described in Lemma 11 allow to compute $\rho_{y,x^-,x^+}(D+1)$ for every $y \in \{1, \ldots, D+1\}$ and for every $x^-, x^+ \in \mathcal{P}_{D+1}$ with $x^- \leq x^+$.

- Consider first the case $\Phi_{D+1}(G) \in]x^-, x^+[$. We necessarily have $\rho_{y,x^-,x^+}(D+1) = \rho_{y,x^-,x^+}(D)$ because we cannot start a new plateau since $x^- < \Phi_{D+1}(G) < x^+$.

- Assume that $\Phi_{D+1}(G) = x^-$ and $x^- < x^+$. We cannot start a new plateau because $x^- < x^+$. Thus we have to find the best y plateaus such that the lower bound is at least $\Phi_{D+1}(G) = x^-$ and at most x^+. We get that $\rho_{y,x^-,x^+}(D+1) = \min_{x \in \mathcal{P}_{D+1}, x^- \leq x \leq x^+}(\rho_{y,x,x^+}(D) + x - x^-)$.
- If $\Phi_{D+1}(G) = x^+$ and $x^- < x^+$, then it is similar than the previous case (symmetric case).
- Consider the case $x^- = x^+ = \Phi_{D+1}(G)$. It is possible to continue the current plateau or it is possible to start a new one. In the first case, the score is $\rho_{y,x^-,x^+}(D)$. In the second case, the score is minimum score among all the optimal solutions composed of $y-1$ plateaus, that is $\min_{x,x' \in \mathcal{P}_{D+1}, x \leq x'}(y-1, x, x')$. Thus, $\rho_{y,x^-,x^+}(D+1)$ is the minimum among these two scores.
- If $\Phi_{D+1}(G) < x^-$ or $\Phi_{D+1}(G) > x^+$, then there is no admissible solution and, by convention, we have $\rho_{y,x^-,x^+}(D+1) = \infty$.

□

Our dynamic programming algorithm computes for every $D \in \{1, \ldots, D_G\}$, for every $y \in \{1, \ldots, D\}$ and for every $x^-, x^+ \in \mathcal{P}_D = \{\Phi_1(G), \Phi_2(G), \ldots, \Phi_D(G)\}$ with $x^- \leq x^+$, the optimal solution $\rho_{y,x^-,x^+}(D)$ for the sub-problem defined by the set $\Phi_1(G), \Phi_2(G), \ldots, \Phi_D(G)$, a number y of plateaus, and such that $x^- = \min_{x \in I_y \cap \mathbb{N}} x$ and $x^+ = \max_{x \in I_y \cap \mathbb{N}} x$ (I_y is the last plateau of the optimal solution for this sub-problem, that is the current plateau if we consider the original instance of the plateau problem). In other words, $\rho_{y,x^-,x^+}(D)$ is the optimal solution for the sub-problem induced by the first D values of Φ, such that the number of plateaus is y and the last plateau has minimum value x^- and maximum value x^+. If there is no admissible solution, we set $\rho_{y,x^-,x^+}(D) = \infty$.

To prove Theorem 18, we first prove in Lemma 11 the correctness of the computation of $\rho_{y,x^-,x^+}(D+1)$.

Lemma. 11. *First, $\rho_{1,\Phi_1(G),\Phi_1(G)}(1) = 0$. For every $D \in \{1, \ldots, D_G - 1\}$, for every $y \in \{1, \ldots, D+1\}$, for every $x^-, x^+ \in \mathcal{P}_{D+1}$ with $x^- \leq x^+$:*

- *If $\Phi_{D+1}(G) \in]x^-, x^+[$, then $\rho_{y,x^-,x^+}(D+1) = \rho_{y,x^-,x^+}(D)$;*
- *If $\Phi_{D+1}(G) = x^-$ and $x^- < x^+$, then $\rho_{y,x^-,x^+}(D+1) = \min_{x \in \mathcal{P}_{D+1}, x^- \leq x \leq x^+}(\rho_{y,x,x^+}(D) + x - x^-)$;*
- *If $\Phi_{D+1}(G) = x^+$ and $x^- < x^+$, then $\rho_{y,x^-,x^+}(D+1) = \min_{x \in \mathcal{P}_{D+1}, x^- \leq x \leq x^+}(\rho_{y,x,x^+}(D) + x^+ - x)$;*
- *If $x^- = x^+ = \Phi_{D+1}(G)$, then $\rho_{y,x^-,x^+}(D+1) = \min(\rho_{y,x^-,x^+}(D), \min_{x,x' \in \mathcal{P}_{D+1}, x \leq x'}(y-1, x, x'))$;*
- *If $\Phi_{D+1}(G) < x^-$ or $\Phi_{D+1}(G) > x^+$, then $\rho_{y,x^-,x^+}(D+1) = \infty$ (there is no solution).*

Proof of Lemma 11. We prove the result by induction. Clearly, $\rho_{1,\Phi_1(G),\Phi_1(G)}(1) = 0$.

Assume that we have computed $\rho_{y,x^-,x^+}(D)$ for every $D \in \{1, \ldots, D'\}$, for every $y \in \{1, \ldots, D\}$, for every $x^-, x^+ \in \mathcal{P}_D$ with $x^- \leq x^+$. We prove that the five cases described in Lemma 11 allow to compute $\rho_{y,x^-,x^+}(D+1)$ for every $y \in \{1, \ldots, D+1\}$ and for every $x^-, x^+ \in \mathcal{P}_{D+1}$ with $x^- \leq x^+$.

- Consider first the case $\Phi_{D+1}(G) \in]x^-, x^+[$. We necessarily have $\rho_{y,x^-,x^+}(D+1) = \rho_{y,x^-,x^+}(D)$ because we cannot start a new plateau since $x^- < \Phi_{D+1}(G) < x^+$.

- Assume that $\Phi_{D+1}(G) = x^-$ and $x^- < x^+$. We cannot start a new plateau because $x^- < x^+$. Thus we have to find the best y plateaus such that the lower bound is at least $\Phi_{D+1}(G) = x^-$ and at most x^+. We get that $\rho_{y,x^-,x^+}(D+1) = \min_{x \in \mathcal{P}_{D+1}, x^- \leq x \leq x^+}(\rho_{y,x,x^+}(D) + x - x^-)$.

- If $\Phi_{D+1}(G) = x^+$ and $x^- < x^+$, then it is similar than the previous case (symmetric case).

- Consider the case $x^- = x^+ = \Phi_{D+1}(G)$. It is possible to continue the current plateau or it is possible to start a new one. In the first case, the score is $\rho_{y,x^-,x^+}(D)$. In the second case, the score is minimum score among all the optimal solutions composed of $y-1$ plateaus, that is $\min_{x,x' \in \mathcal{P}_{D+1}, x \leq x'}(y-1, x, x')$. Thus, $\rho_{y,x^-,x^+}(D+1)$ is the minimum among these two scores.

- If $\Phi_{D+1}(G) < x^-$ or $\Phi_{D+1}(G) > x^+$, then there is no admissible solution and, by convention, we have $\rho_{y,x^-,x^+}(D+1) = \infty$.

□

By Lemma 11, the score of an optimal solution for every number $y \in \{1, \ldots, D_G\}$ of plateaus is:

$$\min_{x^-,x^+ \in \mathcal{P}_D, x^- \leq x^+} \rho_{y,x^-,x^+}(D_G).$$

We finally prove the time-complexity of our dynamic programming algorithm in Lemma 12.

Lemma. 12. *The time complexity of the algorithm of Lemma 11 is $O(D_G^5)$.*

Proof of Lemma 12. We first consider all the cases but the fourth. For every $D \in \{0, \ldots, D_G - 1\}$, for every $y \in \{1, \ldots, D+1\}$, for every $x^-, x^+ \in \mathcal{P}_{D+1}$ with $x^- \leq x^+$, we have to compute $\rho_{y,x^-,x^+}(D+1)$. There are $O(D_G^4)$ such computations. All the cases (but the fourth), can be calculated in $O(D_G)$ time. Thus, we get the $O(D_G^5)$-time complexity.

Now consider the fourth case in which $x^- = x^+$. Thus, for every $D \in \{0, \ldots, D_G - 1\}$, for every $y \in \{1, \ldots, D+1\}$, for every $x^- = x^+ \in \mathcal{P}_{D+1}$, we have to compute $\rho_{y,x^-,x^+}(D+1)$. There are $O(D_G^3)$ such computations. Furthermore, $\rho_{y,x^-,x^+}(D+1) = \min(\rho_{y,x^-,x^+}(D), \min_{x,x' \in \mathcal{P}_{D+1}, x \leq x'}(y-1, x, x'))$ can be computed in $O(D_G^2)$ time. Thus, we get the $O(D_G^5)$-time complexity. □

6.4 Hierarchical Plateaus

As a final interpretation of role of D, we perform a hierarchical construction of plateaus, in a spirit analogous to the construction of a dendogram. In hierarchical agglomerative clustering (such as maximum/minimum/average linkage), recall that at any step, the two clusters merged are those whose distance is minimized. Likewise, consider two consecutive plateaus, each consisting of a set of values $\{(D, \Phi_D)\}$. As a measure of the coherence of all values (D, Φ_D) for the union of these two plateaus, one may compute the maximum difference between any two values $\Phi_\cdot$ on these plateaus. Merging the two plateaus realizing the minimum value then yields a dendogram.

More formally, we construct the rooted tree $T = (V, E)$ representing the hierarchical plateaus as follows. There is a leaf per possible value of D and there are $D_G - 1$ internal nodes (including the root). Let $L = \{v_1, \ldots, v_{D_G}\}$ be the set of leaves of T and let $U = \{u_1, \ldots, u_{D_G - 1}\}$ be the set of internal nodes of T. Let $V = L \cup U$. Algorithm 2 formally describes the set of edges of T.

Algorithm 2 Construction of the tree representing the hierarchical plateaus.

Require: The values $(\Phi_1(G), \ldots, \Phi_{D_G}(G))$.
The set of nodes is $V = L \cup U$, where $L = \{v_1, \ldots, v_{D_G}\}$ and $U := \{u_1, \ldots, u_{D_G-1}\}$.
$d := D_G$, $x := 1$, $E := \emptyset$.
Let $(I_1, I_2, \ldots, I_d)$ be the $d = D_G$ initial plateaus each composed of 1 point.
We associate a leaf node to every initial interval: $v(I_i) = v_i$ for every $i \in \{1, \ldots, d\}$.
while $d \geq 2$ **do**
  $i' := \arg\min_{i \in \{1,\ldots,d\}} \max_{D,D' \in I_i \cup I_{i+1}} \Phi_{D'}(G) - \Phi_D(G)$.
  $E := E \cup \{v(I_{i'}), u_x, v(I_{i'+1}), u_x\}$.
  $I_{i'} := I_{i'} \cup I_{i'+1}$ and $I_j := I_{j+1}$ for every $j \in \{i'+1, d-1\}$.
  $v(I_{i'}) = u_x$ and $v(I_j) = v(I_{j+1})$ for every $j \in \{i'+1, d-1\}$.
  $d := d - 1$, $x := x + 1$.
return $T = (V, E)$

7 Experiments

We present experiments in three directions:

- In Section 7.2, we compare a clustering and an edited version of it, in order to assess the value of D required to counter-balance the magnitude of edits.
- In Section 7.3, we investigate the role of D to compare clusterings yielded by `k-means++` and recover the *right* number of clusters when the number of clusters passed to `k-means++` is too large.
- In Section 7.4, we argue that a normalized score associated with our matchings provides more stable information than the information theoretical measure variation of information (VI).

7.1 Implementation

Generic code. We implemented a version of Algorithm 1, which takes as input a graph $G = (V, E, w)$ and a diameter D. This implementation is integrated to the *Core / Combinatorial algorithms and data structures* (CADS) component of the Structural Bioinformatics Library (`http://sbl.inria.fr`). Documentation can be accessed directly from `http://sbl.inria.fr/doc/D_family_matching-user-manual.html`. A Jupyter notebook accompanies this implementation to help users get started. As it may be noticed, the main class T_Spanning_tree_solver which is a direct implementation of Algorithm 1, is templated by an intersection graph type and **(i)** a spanning tree generator $\mathcal{R}$, **(ii)** a stop condition ($\Pi\mathcal{M}$) and **(iii)** an algorithm $\mathcal{A}$.

Instantiation for our experiments. For the following experiments, we use an instantiation of the previous generic algorithm, $STS(G, D)$ which has the following ingredients: **(i)** the spanning tree generator $\mathcal{R}$ returns a *maximum spanning tree*, or a *random spanning tree*; **(ii)** the property $\Pi(\mathcal{M})$ returns true once we have computed a solution on the maximum spanning tree, as well as a solution on $n_i = (10,000)$ distinct random spanning trees (for a given n_i); **(iii)** $\mathcal{A}$ is the algorithm described in Theorem 9 (Section 4) with an additional step: the addition of edges whose endpoints belong to the same meta-cluster – see Remark 1. The solution returned for a given graph G and a diameter D is the best yielded by the aforementioned $1 + n_i$ spanning trees.

The corresponding executable from the Structural Bioinformatics Library is sbl-d-family-matching.exe. Individual running times (< one minute on a laptop computer) are not further analyzed.

7.2 Experiments on random and edited clusterings

Rationale. We test our algorithm on pairs of clusterings (F, F'), with F a random clustering, and F' an edited version of F. The goal is to assess the ability of our algorithm to retrieve matchings such as the one of Figure 1, stressing the role of parameter D.

Dataset: random clusterings. The number of clusterings of a set Z of size t into r clusters is the number of distinct partitions of this set into r nonempty subsets. Its number is the Stirling number of second kind [22]. Adding up all these numbers yields the number of partitions of the set Z into any number of subsets, which is the Bell number $B(t)$ [17]. Such clusterings were generated using a Boltzmann sampler [15, Example 5]. Since clustering usually aims at grouping data points into a relatively small number of clusters, two pairs of parameters were used ($t = 1\ 000, r = 20$) and ($t = 3\ 000, r = 50$). Both yielded similar results, so that we only report on the former. Due to the randomness, we perform a number N_r of repetitions. Practically, we use $N_r = 10$ times for each pair (t, r).

Dataset: edited clusterings. We build random pairs of clusterings (F, F') by copying F into F' and editing F', in two steps. First, we perform e union operations to reduce the number of clusters to $r - e$. Second, the elements of the remaining clusters are jittered: for each cluster, a fraction τ of its items are distributed amongst the remaining $k - 1$ clusters uniformly at random. Practically, we take $e \in \{0, \lfloor r/4 \rfloor, \lfloor r/2 \rfloor\}$ and $\tau \in \{0.05, 0.1, 0.2\}$. Note that for $e = 0$, F' is a jittered version of F (i.e. the numbers of clusters are identical). Denote $\#(t, r)$ the number of pairs (t, r) used, and similarly for $\#e$ and $\#\tau$. Summarizing, this setup yields $N_r \times \#(t, r) \times \#e \times \#\tau = 180$ comparisons, which are ascribed to 9 scenarios (3 values for $e \times 3$ values for τ) denoted $EeJy$, where $y = 100\tau$.

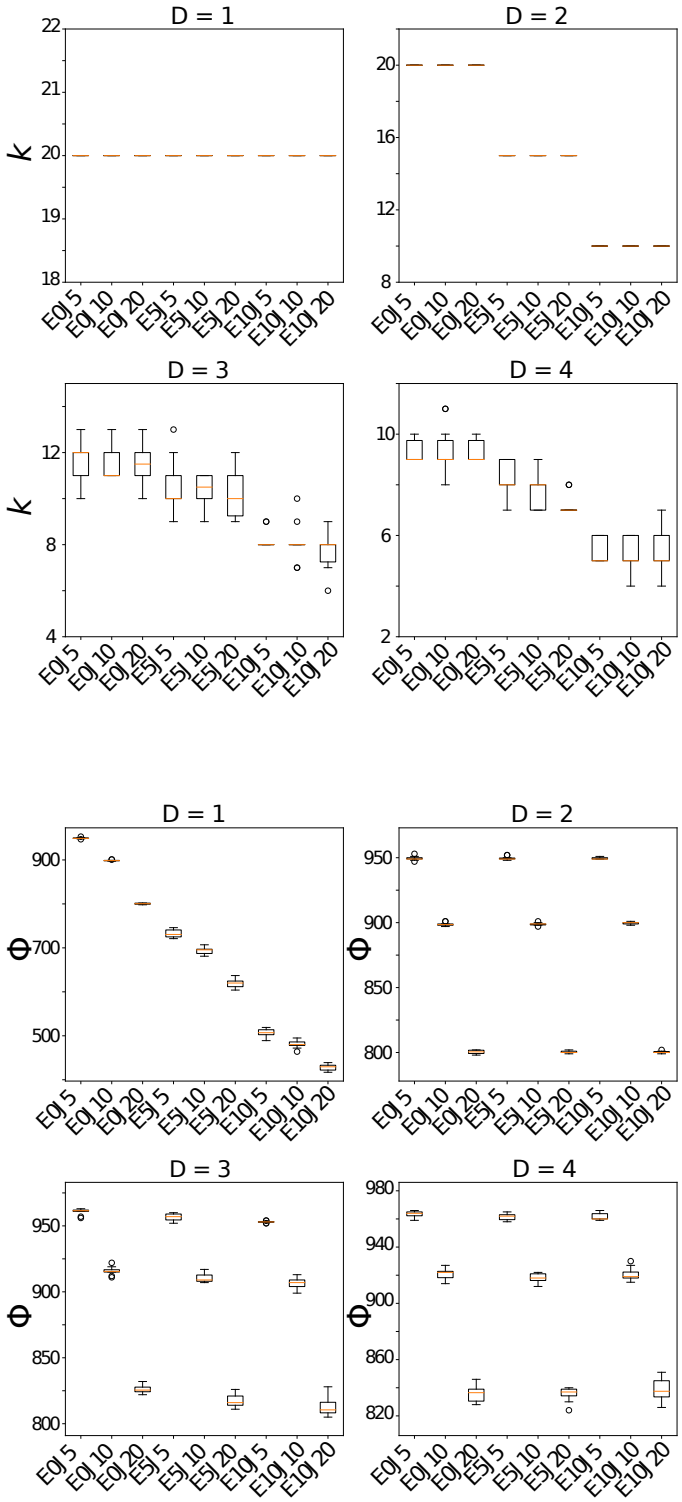
Statistics. These 180 comparison pairs are fed to algorithm $STS(G, D)$ for $D \in \{1, 2, 3, 4\}$. Since each protocol is repeated $N_r = 10$ times, we report a moustache plot of the score Φ as well as the number of meta-clusters k, collected over the N_r repeats (for each value of D).

Results. For $D = 1$ (left panel, top left), our algorithm always outputs $k = 20$ meta-clusters. On closer inspection, this is due to the policy with regards to singletons. That is, our algorithm with $D = 1$ returns a matching (meta-clusters involving two clusters) plus singletons (meta-clusters with a single cluster; note that these do not affect the score $\Phi(\mathcal{S})$). After removing these singletons, we do get the correct number of meta-clusters (20, 15, and 10). For $D = 2$ (left panel, top right), as expected, our algorithm recovers the correct number k of meta-clusters (20, 15, and 10) for each comparison scenario ($e = 0$, $e = 5$, and $e = 10$ fusions). The returned scores (right panel, top right) confirm that our algorithm matches the merged clusters in F' with their split counterparts in F at any jitter level.

This is made clear by comparing scores for scenarios in which we perform fusion operations (E5 or E10) to the ones where we do not (E0). Across all these scenarios, at an equivalent jitter level, the scores are nearly identical. Moreover, the fact that for $D = 1$, the score (right panel, top left) decreases linearly with respect to the number of fusions bolsters this hypothesis.

For $D = 3$, Φ is comparable to previous case ($D = 2$). The situation for k is more contrasted. We still identify three plateaus corresponding to the three different fusion scenarios. The scenarios in which we perform no fusions (E0) (resp. 5, E5, and 10, E10) tend to have a k which oscillates

Figure 7 Algorithm $STS(G, D)$ for clusterings with $(t = 1\ 000, r = 20)$. **(Left panel)** Best value for k as a function of the 9 scenarios. **(Right panel)** Scores $\Phi_D(\cdot)$ as a function of the 9 scenarios.



around 12 (resp. 10 and 8). The jitter seems to have little effect. For $D = 4$, we notice a similar behavior as for $D = 3$ but with different k-values. These results prompted the strategy for the choice of D presented in Section 6 as we can see that increasing $D \geq 2$ yields near identical scores but merges the clusters.

Conclusion. These experiments illustrate that our algorithm behaves as expected:

- On the one hand, scores are remarkably stable with respect to unions, as unions are retrieved within meta-clusters.
- On the other hand, the scores continuously degrade as a function of the jitter level, which is also expected since in general the items which have been shuffled cannot be recovered.

7.3 On the separability of clusters and the role of D

Rationale. We assess the role of D to provide insights of the *correct* number of clusters in a dataset. In other words, we provide a quantitative assessment on the following observation [41]: "*In fact, the* `right` *number of clusters in a dataset often depends on the* `scale` *at which the dataset is inspected*". Along the way, we compare the results to those yielded by the gap statistic [42].

Datasets. We generate three point clouds composed of five random samples each drawn from a $2D$ Gaussian distribution. The relative position of the Gaussians is determined by a distance parameter d controlling the separability of the four clusters associated with the five random samples (Figure 8). Practically, we use $d = 5, 20, 50$. We then use `k-means++` on these three point clouds, with two values: $k_1 = 20$ and $k_2 = 50$ (k_i refers to the k parameter of `k-means++`). Since both these values yield an over-segmentation of the datasets, we challenge our method to retrieve the segmented clusters, using in particular plateaus plots (Section 6).

While these are 2D cases for the sake of exposure, our machinery naturally applies in high-dimensional spaces where inferring the structure of a clustering is much more challenging.

Results. Three scenarios can be identified:

- **The dataset is not separable beyond the connected components of the intersection graph.** This is an easy case since the score Φ_D reaches the maximum possible value when meta-clusters correspond to connected components of the intersection graph (Figure 9 (A); $D = 8$ and $k = 4$ meta-clusters).
- **Dataset is separable: the plateaus strategy yields an unambiguous choice for D.** The best choice for D stems from the analysis of plateaus increments (Def. 11; Figure 11 (B), $D = 8$ yields $k = 3$ meta-clusters).
- **The dataset is not separable: no clear choice for D.** In this case, each successive plateaus has a significant increment and there is no clear break (Figure 9(C)).

Gap statistic. The gap statistic performs comparably to our algorithm (Figure 10). However, it requires a reference distribution obtained via randomization, so that the number of clusters returned may be subject to variation.

Conclusion. Summarizing, the plateaus based analysis of scores provides insights on the plausible number of clusters.

Remark 3. On the sensitivity to outliers.

When clustering with `k-means++` *the assignment of outliers to the different clusters is inherently unstable. When comparing two such clusterings, this creates edges with small weights in the intersection graph. For large values of D, these edges trigger the coalescence of meta-clusters. As a heuristic, such edges may be pruned from the intersection graph – a strategy not used in our experiments.*

Figure 8 Parameterized dataset defined from a mixture of five Gaussians. **(A)** The distance parameter d controls the relative position of the five Gaussian blobs. The covariance matrix of the Gaussians is provided in the figure. **(B, C, D)** Random samples of $t = 5,000$ points for $d = 50, 20, 5$ respectively. Four regions/clusters are well separated for large values of d. Each point random sample was clustered using `k-means++` ($k = 5$).

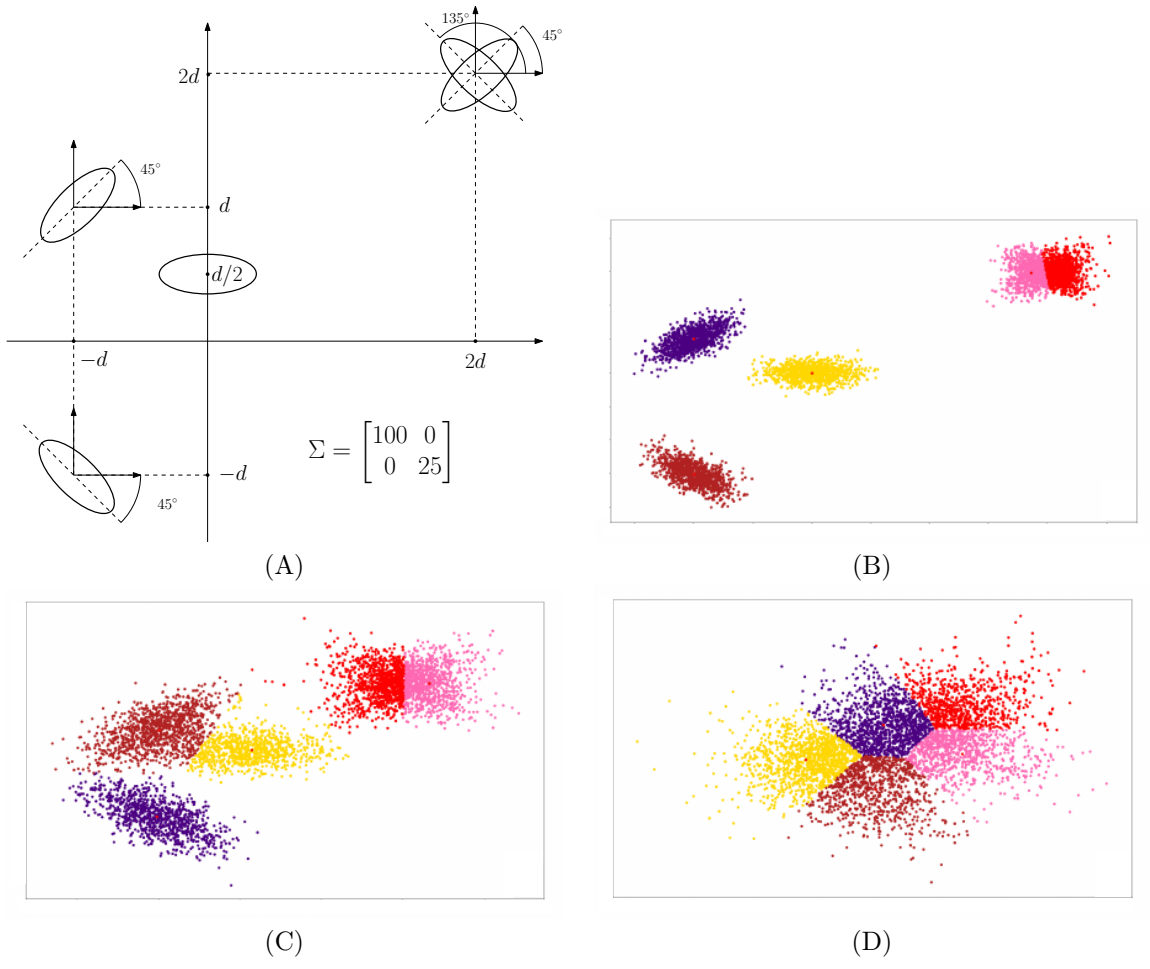
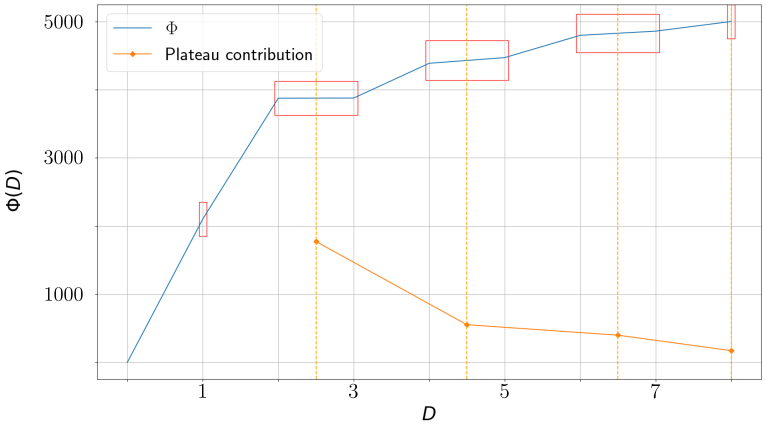
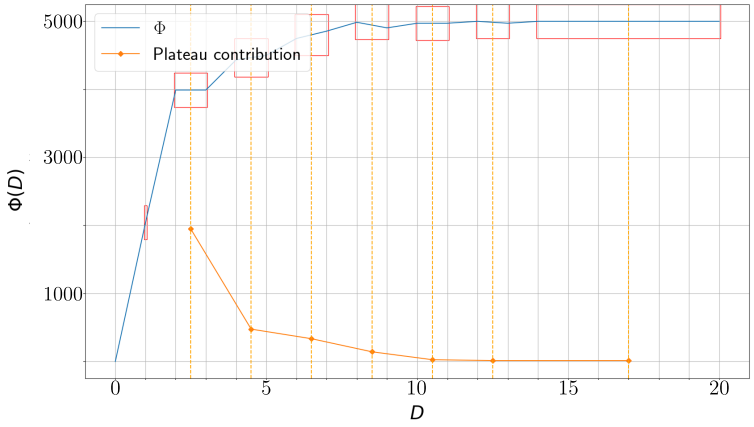


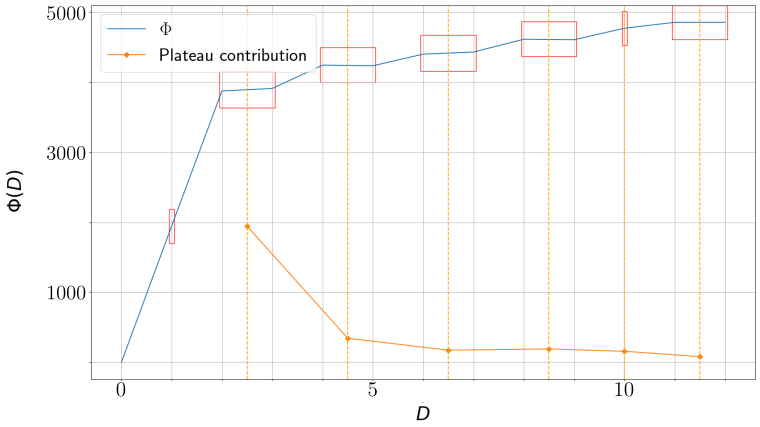
Figure 9 The plateaus plots for the three data sets–see text for details. (A) $d = 50$, $k = 4$ meta-clusters suggested for $D = 8$. **(B)** $d = 20$, $k = 3$ meta-clusters suggested for $D = 8$. **(C)** $d = 5$ No obvious choice for the number of meta-clusters.



(A)

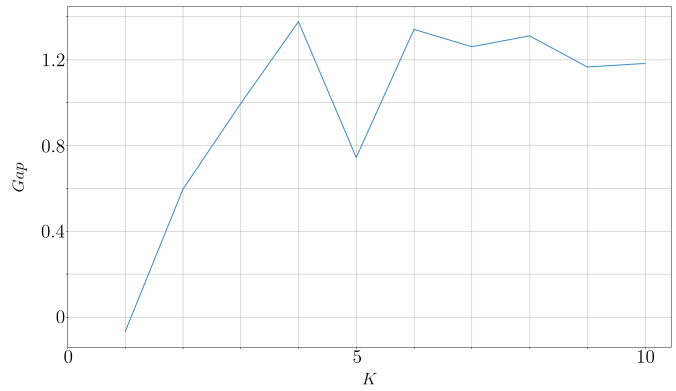


(B)

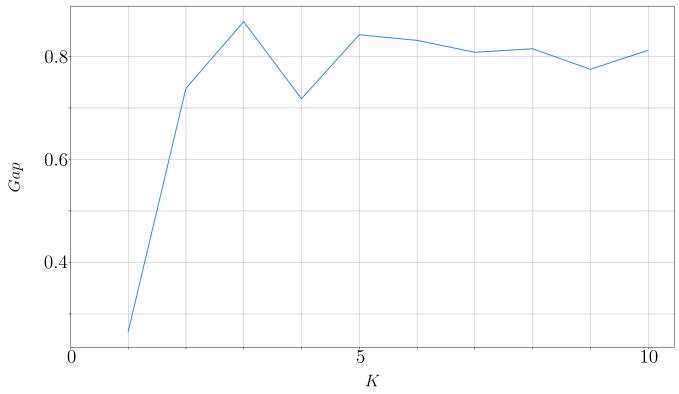


(C)

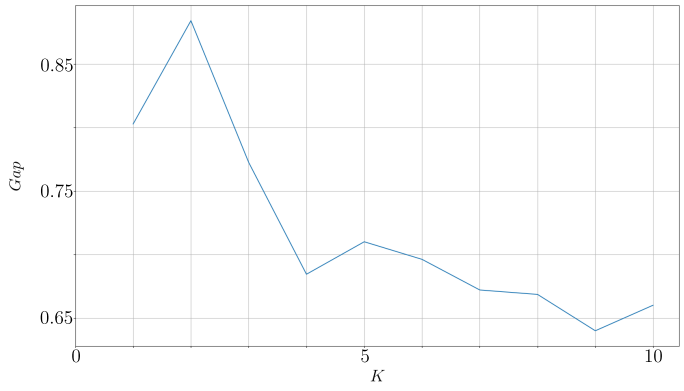
Figure 10 The gap statistic from [42] for the three data sets. (A) d=50; the maximum value of the gap statistic hints at 4 clusters, with a comparable value for 6 clusters. **(B)** d=20; 3 clusters suggested, with a comparable value for 5 clusters. **(C)** d=5; 2 clusters suggested.



(A)



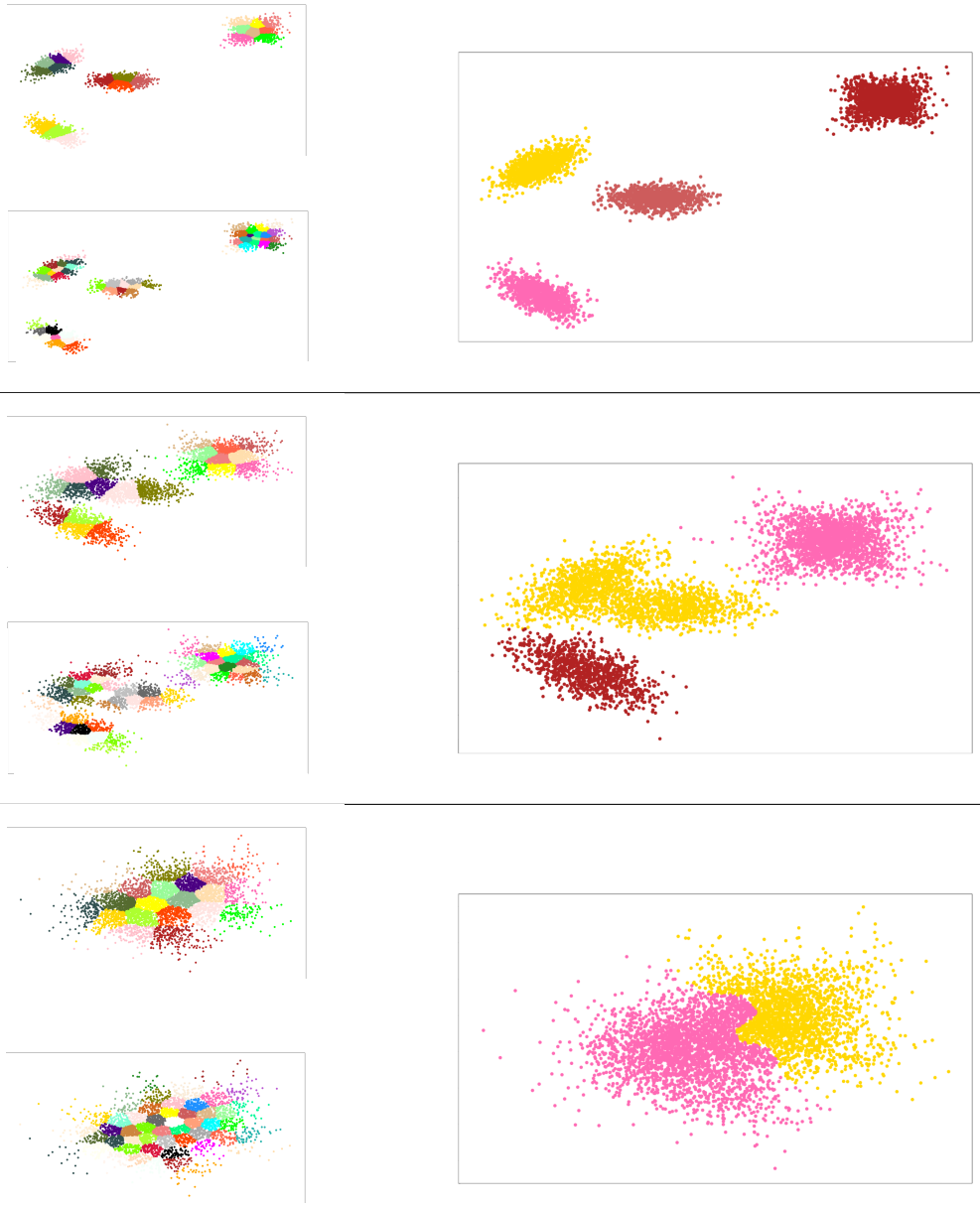
(B)



(C)

d50

Figure 11 The meta-clusters for the three data sets. Left column: the two clusterings compared; right column: meta-clusters **(Top)** $d = 50$: two of the five Gaussians have merged and are separated from the other three– a data set which is not separable beyond the connected components of the intersection graph. The plateau plot suggests $D = 8$ and 4 meta clusters (Section 7.3). **(Middle)** $d = 20$: the 5 Gaussians define a dataset that may be separated into four connected components. The plateau plot suggests $D = 8$ and 3 meta clusters (Section 7.3). **(Bottom)** $d = 5$: the data set is not separable. The plateau plot does not suggest any specific number of meta clusters(Section 7.3).



7.4 Comparison to the Variation of Information (VI)

Rationale. As noticed in the conclusion of Section 7.2, edited clustering can be used to study the sensitivity of cluster analysis methods to merges and shuffles. We therefore compare the Variation of Information [32] against our method in this respect. More specifically, consider the normalized variation of information VI defined as $s_{VI} = \mathrm{VI}/\log t$ against our normalized score $s_\Phi = 1 - \Phi_D(\cdot)/t$. Recall that t is the number of points. To compare s_{VI} and s_Φ, we resort to scatter plots of the values as well as the standards deviations, for different edit and jitter scenarii, with a focus on (i) the relative values of s_{VI} vs s_Φ, and (ii) the stability upon increasing edits and jitter level.

Dataset. We use the dataset from Section 7.2.

Results. Focusing on s_{VI} and s_Φ, our analysis relies on scatter plots of values (Figure 12) and standard deviations at fixed jitter levels (Figure 13) and fixed number of unions (Figure 14). We use different symbols depending on the considered scenario; the copy number of a symbol represents the number of repeats.

For $D = 1$, s_Φ is smaller than VI only in scenarios with no union operations. This is expected as our algorithm returns a perfect matching, so that unions are detrimental to the score.

For $D > 1$, we note several key differences with VI:

- s_Φ is always smaller than s_{VI} (Figure 12);
- s_Φ is more robust since $\sigma(s_\Phi) < \sigma(s_{VI})$ (Figs. 13 and 14).
- s_Φ is remarkably stable against merges, as evidenced by two facts. First, the standard deviation at any fixed jitter level is always very close to 0 (Figure 13). This stability is not observed for s_{VI}. Second, the jitter level has the same effect irrespective of the union scenario (Figure 14).

Conclusion. The variation of information, which is a global measure, is sensitive to cluster edits (merges, splits). On the opposite, the ability of our method to identify merges and splits makes it more suitable when insights on correspondences between clusterings are sought.

Figure 12 Normalized score s_{VI} versus normalized score s_Φ of algorithm $STS(G, D)$. See text for definitions. Each marker is a different union scenario and each color represents a different jitter scenario following the legend on the upper right. We plot the $y = x$ function for reference.

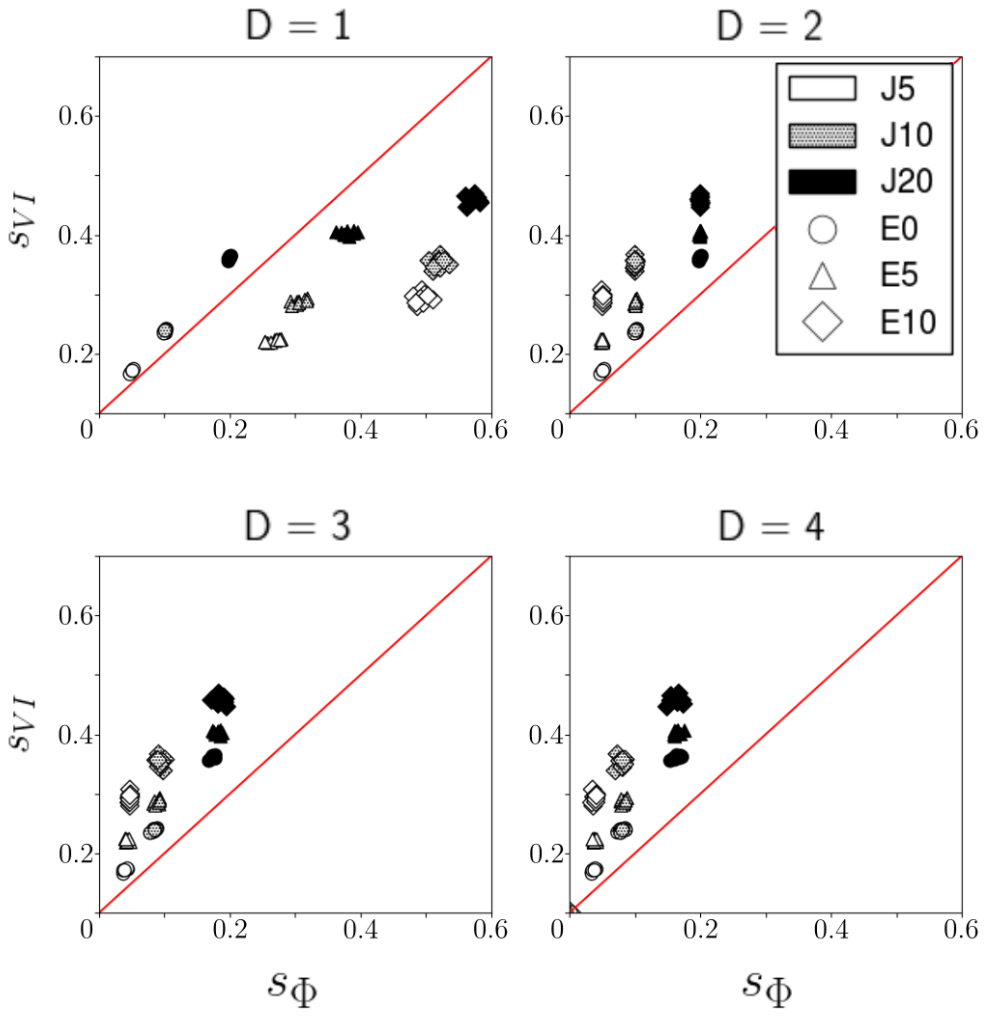


Figure 13 σ **of normalized score** s_{VI} **versus** σ **of normalized score** s_Φ **of algorithm** $STS(G, D)$ **with respect to jitter levels (i.e. experiments corresponding to all edits aggregated).** See text for definitions. Each color represents a different jitter scenario following the legend on the upper right. We plot the $y = x$ function for reference.

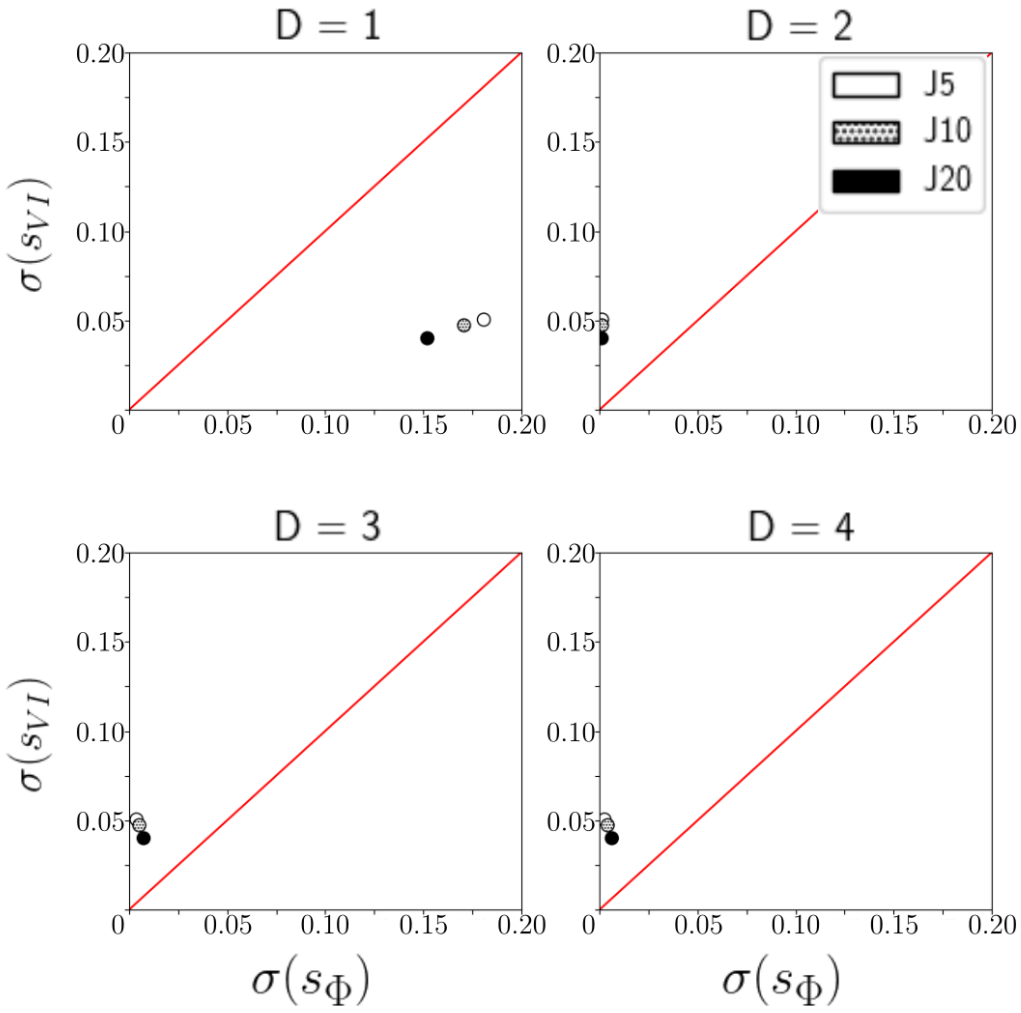
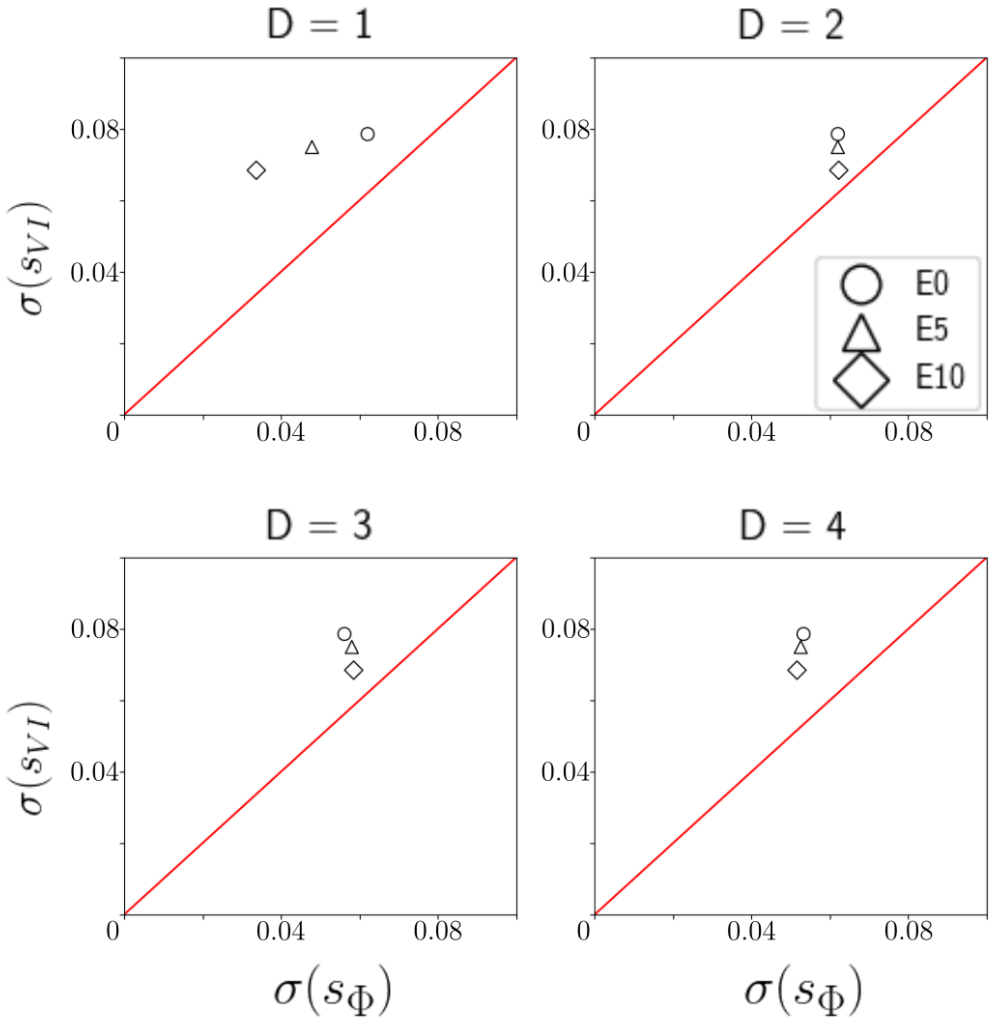


Figure 14 σ **of normalized score** s_{VI} **versus** σ **of normalized score** s_Φ **of algorithm** $STS(G, D)$ **with respect to number of edits (i.e. experiments corresponding to all jitters aggregated).** See text for definitions. Each marker represents a different union scenario following the legend on the upper right. We plot the $y = x$ function for reference.



8 Conclusion

This paper contributes to a new tier of algorithms to compare two clusterings, based on the identification of groups of clusters matching one-another. This problem had only been considered very recently [20], where a many-to-many clusters correspondence between the unions of two groups of clusters is shown to be computable in polynomial time (in the number of clusters) using submodular optimization techniques. We instead target the problem of reporting an arbitrary number of meta-clusters. While the information obtained is finer than two union of cluster corresponding to one another, the endeavor has a price. Our problems are indeed proved to be hard for general bipartite graphs (even if the maximum degree is at most three), with however polynomial time dynamic programming algorithms for specific graphs (in particular trees). These algorithms can in turn be used to design efficient algorithms, based on spanning trees, for general graphs.

From a practical standpoint, experiments illustrate several key features of our algorithms. First, the meta-clusters obtained are highly effective to identify splits and merges between clusters. This ability yields a marked improvement when comparing two clusterings, with respect to global methods such as the variation of information, which are very sensitive to such edits. Second, in a manner analogous to the elbow method or variants, the stability of scores associated to meta-clusters, as a function of the diameter parameter D, provides a novel method to suggest the *correct* number of clusters in a clustering. Overall, we anticipate that our algorithms will prove instrumental to identify stable meta-clusters amidst clusterings (from different algorithms, or from the same algorithm with different parameters).

In terms of future work, we foresee three problems of particular importance, which were barely touched upon in previous work. The first one deals with the complexity of the problems we tackle. In the spirit of Lemma 8 (proving the existence of at least one spanning tree T of G such that an optimal solution for the family-matching problem for G constrained by T gives an optimal solution for the family-matching problem for G) we would like to determine the smallest constant κ for which there exists at least one spanning tree T of G such that an optimal solution for the family-matching problem for T (that can be obtained in polynomial time) is a κ-approximation for the family-matching problem for G if G has bounded degree. Furthermore, we conjecture that the D-family-matching problem is not in APX (recall that we proved that it is APX-hard). Note that both conjectures can be true because the existence of the previous tree would not guarantee a polynomial algorithm for determining it.

The second one deals with the design of post-processing algorithms in order to improve the quality of the solutions: e.g. by merging meta-clusters of a given solution or by merging two different solutions. Indeed, in some cases (e.g. complete bipartite graphs), one can easily merge the meta-clusters obtained by our dynamic programming algorithm and obtain a better solution. However, in general, such an improvement is more difficult to compute and the complexity of such algorithms must be investigated.

The third one deals with the stability of meta-clusters. Understanding which assumptions are indeed required to guarantee that our approach yields stable meta-clusters, in particular in terms of separability of the input sample points, would indeed leverage clustering by removing the arbitrariness inherent to the various algorithms and options available.

References

[1] R. Aldahdooh and W. Ashour. DSMK means *density-based split-and-merge k-means clustering algorithm. Journal of Artificial Intelligence and Soft Computing Research*, 3(1):51–71,

2013.

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

[3] J-P. Baudry, A. Raftery, G. Celeux, K. Lo, and R. Gottardo. Combining mixture components for clustering. *Journal of computational and graphical statistics*, 19(2):332–353, 2010.

[4] F. Cazals and R. Tetley. Multiscale analysis of structurally conserved motifs. Submitted.

[5] F. Chataigner, G. Manic, Y. Wakabayashi, and R. Yuster. Approximation algorithms and hardness results for the clique packing problem. *Disc. Appl. Math.*, 157(7):1396–1406, 2009.

[6] F. Chazal, L. Guibas, S. Oudot, and P. Skraba. Persistence-based clustering in riemannian manifolds. *J. ACM*, 60(6):1–38, 2013.

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

[8] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley & Sons, 2006.

[9] A. Cuevas, M. Febrero, and R. Fraiman. Estimating the number of clusters. *Canadian Journal of Statistics*, 28(2):367–382, 2000.

[10] K. Dabrowski, M. Demange, and V. V. Lozin. New results on maximum induced matchings in bipartite graphs and beyond. *Theoretical Computer Science*, 478:33 – 40, 2013.

[11] E. Dahlhaus, D.S. Johnson, C.H. Papadimitriou, P.D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM J. Comput.*, 23(4):864–894, 1994.

[12] A. Dessmark, J. Jansson, A. Lingas, E-M. Lundell, and M. Persson. On the approximability of maximum and minimum edge clique partition problems. *Int. J. Found. Comput. Sci.*, 18(2):217–226, 2007.

[13] S. Dongen. Performance criteria for graph clustering and markov cluster experiments. 2000.

[14] R.G. Downey, V. Estivill-Castro, M.R. Fellows, E. Prieto-Rodriguez, and F.A. Rosamond. Cutting up is hard to do: the parameterized complexity of k-cut and related problems. *Electr. Notes Theor. Comput. Sci.*, 78:209–222, 2003.

[15] P. Duchon, P. Flajolet, G. Louchard, and G. Schaeffer. Boltzmann samplers for the random generation of combinatorial structures. *Combinatorics, Probability and Computing*, 13(4-5):577–625, 2004.

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

[17] P. Flajolet and R. Sedgewick. *Analytic combinatorics*. Cambridge University press, 2009.

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

[19] M. R. Garey and D. S. Johnson. *Computers and intractability*, volume 29. Freeman, 2002.

[20] R. Glantz and H. Meyerhenke. Many-to-many correspondences between partitions: Introducing a cut-based approach. In *SIAM International Conference on Data Mining*, pages 1–9. SIAM, 2018.

[21] O. Goldschmidt and D. S. Hochbaum. A polynomial algorithm for the k-cut problem for fixed k. *Mathematics of operations research*, 19, 1994.

[22] R. Graham, D. Knuth, and O. Patashnik. *Concrete mathematics: a foundation for computer science*. Addison-Wesley, 1989.

[23] C. Hennig. Methods for merging gaussian mixture components. *Advances in data analysis and classification*, 4(1):3–34, 2010.

[24] V. Kann. Maximum bounded 3-dimensional matching is MAX SNP-complete. *Inf. Process. Lett.*, 37(1):27–35, January 1991.

[25] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, 1972.

[26] L. Kaufman and P. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. Wiley, 1990.

[27] K-I. Kawarabayashi and M. Thorup. The minimum k-way cut of bounded size is fixed-parameter tractable. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS*, pages 160–169, 2011.

[28] T. Kodinariya and P. Prashant. Review on determining number of cluster in k-means clustering. *International Journal*, 1(6):90–95, 2013.

[29] B. Larsen and C. Aone. Fast and effective text mining using linear-time document clustering. In *ACM SIGKDD*, pages 16–22. ACM, 1999.

[30] U. Von Luxburg. *Clustering Stability*. Now Publishers Inc, 2010.

[31] P. Manurangsi. Inapproximability of maximum biclique problems, minimum k-cut and densest at-least-k-subgraph from the small set expansion hypothesis. *Algorithms*, 11(1):10, 2018.

[32] M. Meila. Comparing clusterings. 2002.

[33] M. Muhr and M. Granitzer. Automatic cluster number selection using a split and merge k-means approach. In *Database and Expert Systems Application, 2009. DEXA'09. 20th International Workshop on*, pages 363–367. IEEE, 2009.

[34] A. Ng. Clustering with the k-means algorithm. *Machine Learning*, 2012.

[35] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425 – 440, 1991.

[36] R. Rabbany and O. Zaïane. Generalization of clustering agreements and distances for overlapping clusters and network communities. *Data mining and knowledge discovery*, 29(5):1458–1485, 2015.

[37] A. Rodriguez and A. Laio. Clustering by fast search and find of density peaks. *Science*, 344(6191):1492–1496, 2014.

[38] S. Romano, J. Bailey, V. Nguyen, and K. Verspoor. Standardized mutual information for clustering comparisons: one step further in adjustment for chance. In *International Conference on Machine Learning*, pages 1143–1151, 2014.

[39] Y. Rubner, C. Tomasi, and L.J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[40] H. Saran and V. Vazirani. Finding k-cuts within twice the optimal. *SIAM J. Comp.*, 24, 1995.

[41] A. Strehl and J. Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.

[42] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.

[43] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

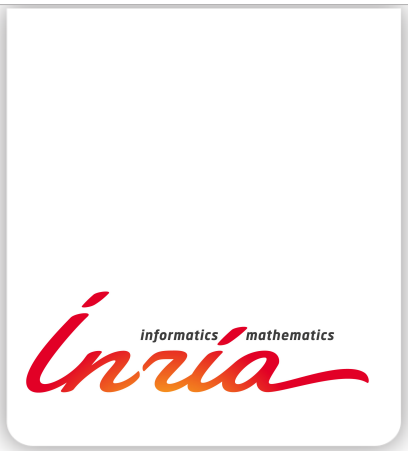
[44] S. Wagner and D. Wagner. Comparing clusterings: an overview. Technical Report 2006-04, 2007.

[45] Q. Xiang, Q. Mao, K. Chai, H. Chieu, I. Tsang, and Z. Zhao. A split-merge framework for comparing clusterings. In *ICML*, 2012.

[46] R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005.

[47] L. Zhao, H. Nagamochi, and T. Ibaraki. *Approximating the Minimum k-way Cut in a Graph via Minimum 3-way Cuts*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[48] D. Zhou, J. Li, and H. Zha. A new mallows distance based metric for comparing clusterings. In *ICML*, pages 1028–1035. ACM, 2005.



RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399