Comparing two clusterings using matchings between clusters of clusters

Frédéric Cazals, Dorian Mazauric, Romain Tetley, Rémi Watrigant

Comparing clusterings using matchings between clusters of clusters

Frédéric Cazals and Dorian Mazauric and Romain Tetley and Rémi Watrigant

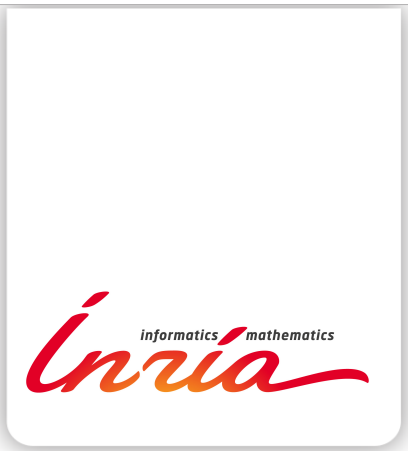
RESEARCH REPORT

N° 9063

August 2017

Project-Team Algorithms-Biology-Structure

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



Comparing 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 — August 2017 — 37 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. Altogether, these pieces of information help assessing the coherence between two clusterings.

More specifically, 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 efficient algorithms, based on spanning trees, for general graphs.

Practically, we illustrate the ability of our algorithms to identify relevant meta-clusters between a given clustering and an edited version of it. By comparing our scores against the Variation of Information, we also show the insights yielded by parameter D.

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 des 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.

Des résultats expérimentaux sont présentés dans deux directions. D'une part, nous montrons comment nos algorithmes peuvent identifier des parties stables entre un clustering et une version *éditée* de celui-ci. D'autre part, nous comparons le score renvoyé par nos algorithmes à une mesure de distance classique pour les clusterings: la variation d'information.

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 [27, 15]. 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 [10]. *k-means and variants* perform a grouping induced by the Voronoi cells of the cluster representatives, which are updated in an iterative fashion [1]. In *density based clustering* methods, a density estimate is typically computed from the data, with clusters associated to the catchment basins of local maxima [4, 5]. Topological persistence may be used to select the significant maxima [3]. 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 [23]. Finally, *spectral clustering* methods define clusters from the top singular vectors of the matrix representing the data (or their similarity) [26]. 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 [1]; 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 [26].

This type of difficulty together with the vast array of clustering schemes actually raises two important questions. The first one deals with the design of cluster quality measures, a topic for which recent work has put emphasis on the performances of spectral clustering methods [17]. The second one, which motivates this paper, is the problem of comparing two clusterings.

Clusterings: comparison and stability assessment. 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.

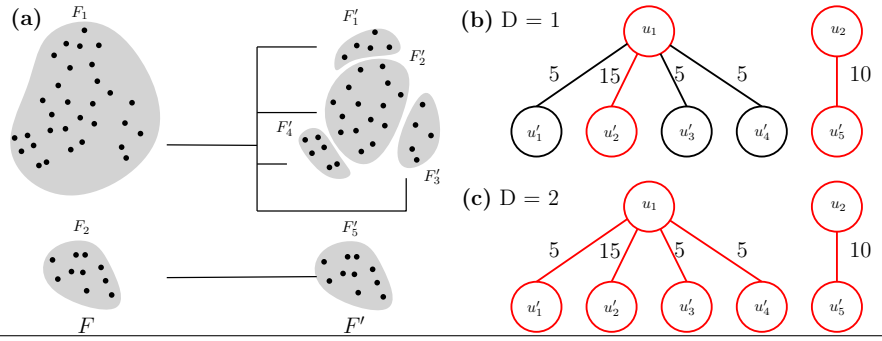
In *set matching based comparisons* [19, 8], a *greedy best effort* 1-to-1 matching between clusters is sought from the contingency table. A statistic is designed by adding up the contributions of these pairs. The resulting measure is often called the *minimal matching distance (MMD)* [20]. To define 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

$$d_{MMD}(F, F') = \frac{1}{t} \min_{\pi \in \Pi} \sum_{i=1}^{t} \mathbf{1}_{F(x_i) \neq \pi(F'(x_i))}, \tag{1}$$

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, Eq. (1) 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. See Figure 1.

In *pair counting methods* [21], 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). A statistic is then devised from these four numbers (e.g. the Rand index). While relevant for problems where pairs are of paramount importance, such methods do not provide any insight on the relationships between 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. Each edge displays the number of shared points by two clusters. In (b) and (c), the intersection graph (Definition 1) is depicted: one node per cluster, an edge between two nodes if the corresponding clusters share at least one point, and the weight of an edge is the number of points shared by the two clusters. Our method 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* [21], the coherence between clusters of F and F' is assessed using the variation of information (VI) between the clusterings. In short, VI is defined from the mutual information between the two clusterings [6], 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.

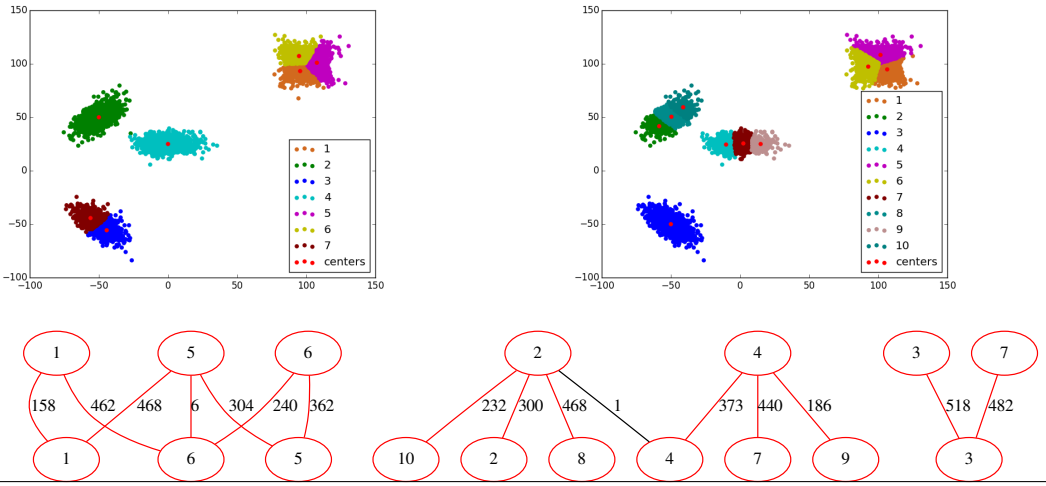
Finally, *optimal transportation based methods* [29] aim at mapping the clusters with one another, and also at accommodating the case of soft clustering. These methods actually rely on the earth mover distance [24]–a linear program which may be seen as a particular case of optimal transportation. In 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 [20]. In particular, the minimal matching distance has been used to study the stability of k-means.

1.2 Main contributions

Rationale. The inability of previous work to go beyond the matching case is clearly detrimental to handle cases where clusters of one clustering can be determined by fusion and/or split operations applied to clusters of the second clustering. See Figure 2 for an illustration. We fill this gap by studying the problem of grouping clusters into meta-clusters. 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

Figure 2 Comparing clusterings to assess the stability of k-means++. 2D point cloud: 5k points drawn according to a mixture of 5 gaussians (the centers of the two in the top right corner are identical, and their principal directions orthogonal). **(Top)** k-means++ with $k = 7$ and $k = 10$, respectively. The clusters are numbered from 1 to k and follow the legends on the right. When the number of centers passed is larger than the exact number of clusters, four here, these clusters get split arbitrarily. **(Bottom)** Meta-clusters (in red) for $D = 3$, superimposed on the so-called *intersection graph* of the two clusterings. For the sake of clarity, we abuse the previous notation by using indices to denote a given vertex ($u_i = i$) so that the vertex labels follow the legends on the top panels. Using these two clusterings, our algorithm recovers the 4 data clusters. Note that this result has been produced with algorithm $STS(G, D)$ with $n_i = 10\,000$ random spanning trees (see Section 6).



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). 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.

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 $D = 2$ solves the case for which one cluster of F corresponds to different smaller clusters of F' (and vice versa). We prove in Lemma 4 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). Finally, when D is finite, it relates to previous work on cut problems on graphs [13, 22, 25, 28]. This is not appropriate for clusterings comparison. Indeed, if the sub-graphs corresponding to meta-clusters have a finite but (too) large diameter, then we cannot *finely* describe the differences between the two clusterings.

Contributions. Our work, which investigates the relationship between two clusterings, shed-

ding 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 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 6, we present extensive experiments, in two directions. First, we illustrate the ability of our algorithms to identify relevant meta-clusters between a given clustering and an edited version of it. Second, by comparing our scores against the Variation of Information, we also show the insights yielded by D.

Due to the lack of space, all the details and proofs can be found in appendix.

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 6 in appendix (Section A) 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.*

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 $i, j \in$*

$\{1, \ldots, k\}$, $i \neq j$, *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.

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

$$\Phi(\mathcal{S}) = \sum_{i=1}^{k} \sum_{e \in E(G[S_i])} w_e. \tag{2}$$

Let $\mathcal{S}_D(G)$ be the set of all D-family-matching for G. We formalize in Definition 3 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 $O(n^2 \log n + nm)$ [12], we deduce the following result.

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

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

In this section, we prove that the D-family-matching problem is NP-complete and that simple strategies can be arbitrarily bad. All the proofs can be found in appendix (Section D).

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. 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. 1. *Let* $D \geq 2$ *be any integer. The decision version of the* D*-family-matching problem is NP-complete for :*

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

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

Given this result, it makes sense to study the approximability of the problem. 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.

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.

Lemma. 3. *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.*

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

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. All the proofs can be found in appendix (Section E). In the following, we explain the main ideas of our exact polynomial time dynamic programming algorithm to solve the D-family-matching problem when the graph is a tree.

Theorem. 2 (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.*

Sketch of the 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.

- For every $i \in \{-1, 0\}$, then

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

- For every $i \in \{1, \ldots, D\}$, then

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

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

For every $v \in V$, the time complexity of the computation of $\Psi_D(v, i)$, for all $i \in \{-1, \ldots, D\}$, 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 $i \in \{-1, \ldots, D\}$, we can deduce an optimal solution $\mathcal{S}$ for the D-family-matching problem for T. Indeed, $\Phi(\mathcal{S}) = \Phi_D(G) = \max_{i \in \{-1,\ldots,D\}} \Psi_D(r, i)$. □

Using similar ideas, we obtained the following results, whose proof can be found in appendix (Section E).

Theorem. 3. *For any $D \in \mathbb{N}^+$, the D-family-matching problem can be solved:*

- *in $O(Dn)$ time if G is a path;*
- *in $O(D^2 n)$ time if G is a cycle(s) or a graph of maximum degree 2.*

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. All proofs can be found in appendix (Section F). This approach relies on computing a solution having a particular structure defined by a given spanning tree T of the input graph. Formally:

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

Definition. 4 (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. 5 (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** $\mathcal{R}(G, t)$ computes the rooted spanning tree T^t of G that is used at step $t \geq 1$ by Algorithm $\mathcal{A}$.
- **An algorithm** $\mathcal{A}(G, T^t, D)$ computes a D-family-matching $\mathcal{S}^t$ constrained by T^t.

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

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. 6 (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.*

Finally, another interesting point of our approach is that it is possible to obtain an efficient (polynomial) heuristic using the dynamic programming algorithm for trees described in Theorem 2. 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 Experiments

6.1 Implementation

We implemented a version of Algorithm 1, which takes as input a graph $G = (V, E, w)$ and a diameter D. This implementation will be integrated shortly as a data analysis application of the Structural Bioinformatics Library (`http://sbl.inria.fr`). The algorithm $STS(G, D)$ (for Spanning Tree Solver) has the following ingredients: **(i)** the spanning tree generator $\mathcal{R}$ returns a *maximum spaning 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 distinct random spanning trees (for a given n_i); **(iii)** $\mathcal{A}$ is the algorithm described in Theorem 2 (Section 4) with an additional step: edges for which both extremities belong to the same meta-cluster (and previously unaccounted for) are added to the said meta-cluster. The solution returned for a given graph G and a diameter D is the best one obtained for the aforementioned $1 + n_i$ spanning trees. Since individual runs took less than one minute on a laptop computer, running times are not further analyzed.

6.2 Experiments on random clusterings

We test our algorithm on pairs of clusterings (F, F'), with F a random clustering, and F' a modified 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.

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 [14]. Adding up all these numbers yields the number of partitions of the set Z into any number of subsets, which is the Bell number $B(t)$ [11]. Such cluterings were generated using a Boltzmann sampler [9, 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 ($t = 3\ 000, r = 50$). Due to the randomness, the process is repeated $N_r = 10$ times for each pair (t, r).

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 $r - e$. Secondly, 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). Summarizing, this setup yields $N_r \times \#(t, r) \times \#e \times \#\tau = 180$ comparisons, which are ascribed to 9 scenarii (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. Due to the lack of space, we only report in Figure 3 and 4 the results for ($t = 1\ 000, r = 20$) processed by $STS(G, D)$. Full details in appendix (Section G).

For $D \leq 2$ (Figure 3, top), 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). For $D = 1$ (Figure 3, top left), this is expected as we should recover a maximum weight matching (perfect for $e = 0$). The jitter level does not compromise this result. For $D = 2$, the returned scores (Figure 4, 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

Figure 3 Algorithm $STS(G, D)$ for clusterings with $(t = 1\ 000, r = 20)$. Best value for k as a function of the 9 scenarii.

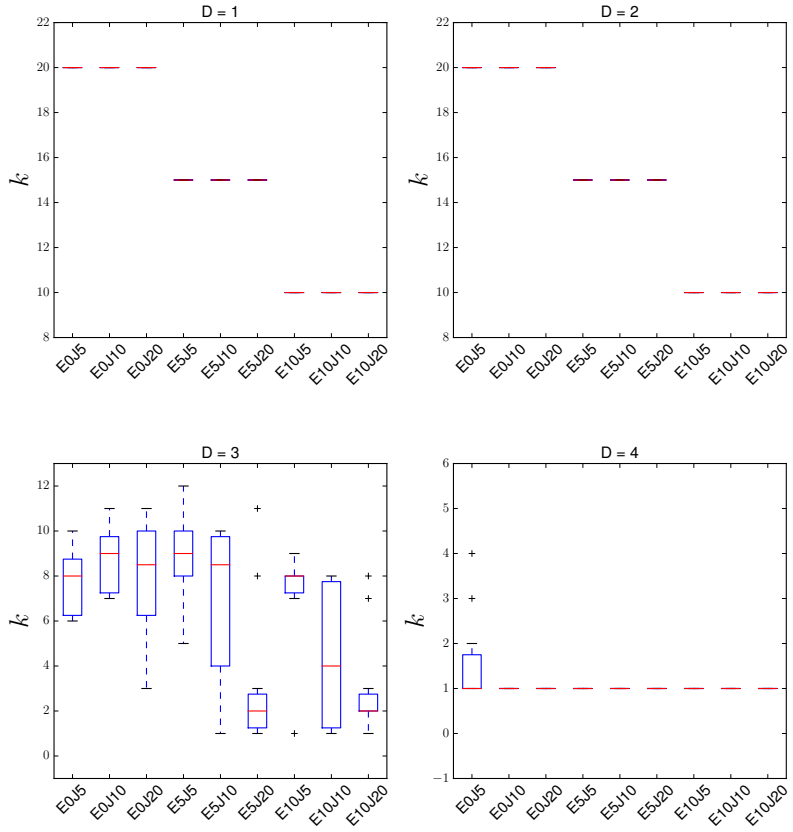


Figure 4 Algorithm $STS(G, D)$ for clusterings with $(t = 1\,000, r = 20)$. Scores s_Φ as a function of the 9 scenarii.

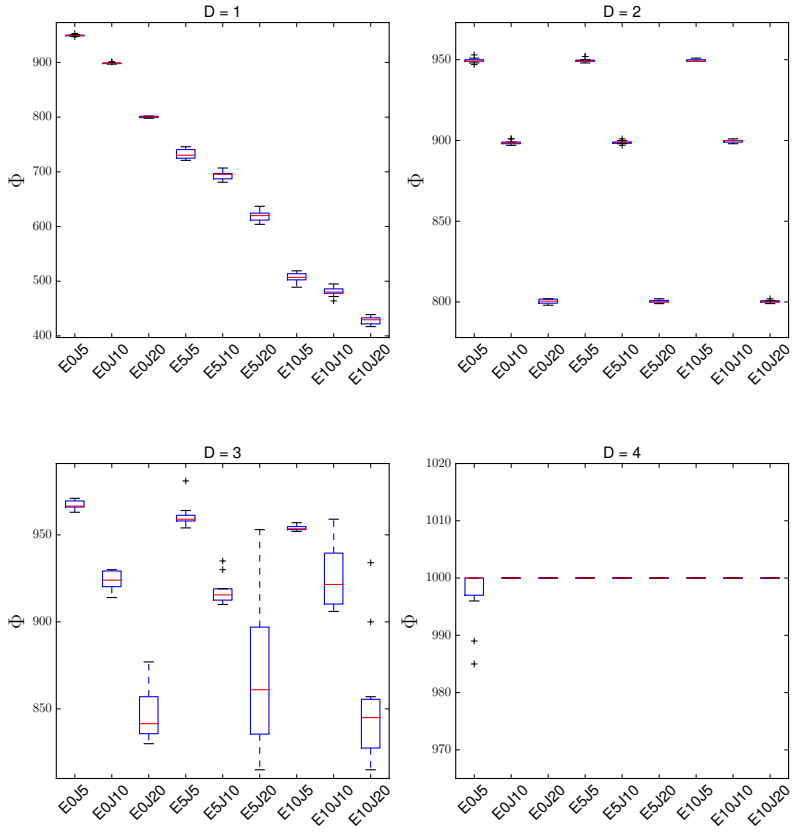
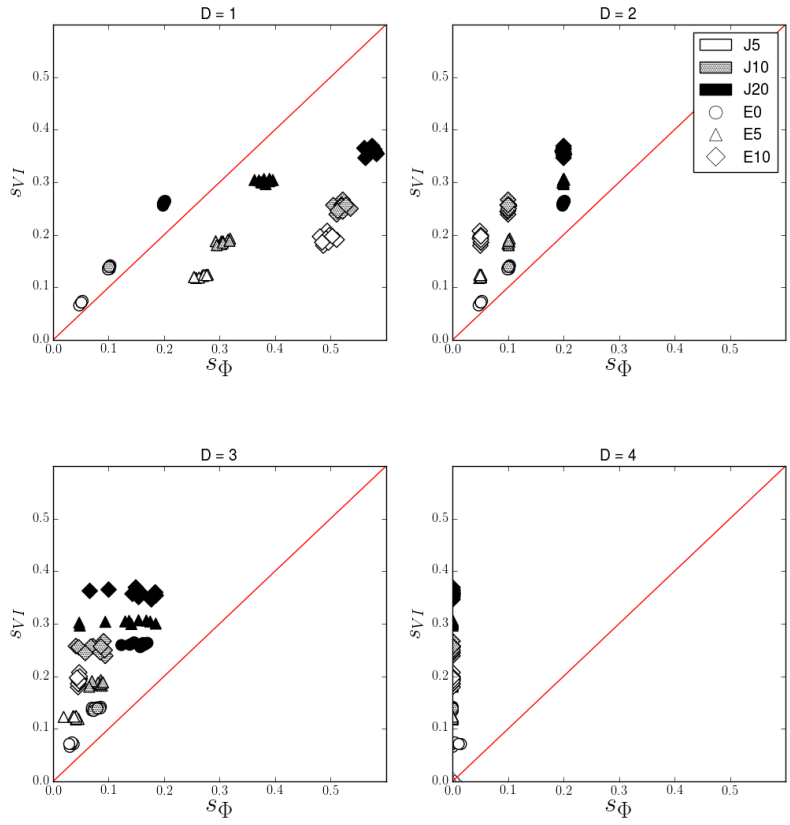


Figure 5 Normalized score s_{VI} versus normalized score s_Φ of algorithm $STS(G, D)$. See text for definitions. Each marker is a different union scenario and each color represents a different jitter scenario following the legend on the upper right. We plot the $y = x$ function for reference.



scenarii in which we perform fusion operations (E5 or E10) to the ones where we do not (E0). Across all these scenarii, at an equivalent jitter level, the scores are nearly identical. Moreover, the fact that for $D = 1$, the score (bottom panel, top left) decreases linearly with respect to the number of fusions bolsters this hypothesis. For $D = 3$, we notice a greater variability in Φ. However the median score stays consistent with the previous case ($D = 2$). We also note that the situation for k is more contrasted. The scenarii in which we perform 5 (E5) and no fusions (E0) tend to have a k which oscillates around 8. As we increase the jitter value (notably for $\tau = 0.2$) and the number of union operations (E10) suddenly k drops closer to 1. For $D = 4$, we no longer discriminate between the different scenarii as our algorithm outputs the full graph as a solution ($k = 1, \Phi = 1\ 000$).

6.3 Comparison to the Variation of Information (VI)

A clustering measure whose properties have been analyzed when clusters are split or merged is the Variation of Information [21]. As opposed to our family matching based strategy, VI does

not build meta-clusters, though. Nevertheless, using the previous datasets, we compare our normalized score $s_\Phi = 1 - \Phi/t$ against the normalized variation of information VI defined as $s_{VI} = \mathrm{VI}/\log t$. We produce a scatter plot of s_Φ against s_{VI}, using different symbols depending on the considered scenario (Figure 5). Note that the copy number of a symbol represents the number of repeats.

For $D \geq 2$, the normalized score s_Φ is consistently smaller than s_{VI}. For $D = 1$, s_Φ is smaller than VI only in scenarii with no union operations. This is expected as our algorithm returns a perfect matching, so that unions are detrimental to the score. For $D = 2$, we note two key differences with VI: s_Φ is always smaller than s_{VI} and s_Φ is robust i.e. constant against union operations. As expected, both s_{VI} and s_Φ are affected by jittering. For $D = 3$, we observe a higher variability in s_Φ, which scales with the amount of jittering and union operations we apply to our clustering. For a high jitter value (black shapes, $\tau = 0.2$), $s_\Phi \in [0.05, 0.2]$ in scenarii where we perform union operations. For $D = 4$, our algorithm outputs the full intersection graph. $s_\Phi = 0$ across all scenarii. In a nutshell: for the correct parameter value ($D = 2$), our algorithm is more consistent and on par with VI.

To conclude, in addition to providing a mapping between meta-clusters, our strategy is less sensitive to jittering for small values of D, and also yields insights on the structure of clusters by increasing D ($D = 4$ vs $D \neq 4$).

6.4 On the choice of D

As seen in the previous examples, large values of D yield trivial values of s_Φ. As a strategy to choose D, we suggest identifying drops in Φ when decreasing D. Indeed, for any range of D corresponding to a *plateau* for Φ, the most significant value for D is the smallest one.

7 Outlook

This paper introduces a new tier of algorithms to compare two clusterings, based on the identification of groups of clusters matching one-another. These problems are 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. In the spirit of Lemma 5 (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 conjecture that there exists at least one spanninng tree T of G such that an optimal solution for the family-matching problem for T (that can be obtained in polynomial time) gives a constant factor approximation for the family-matching problem for G. Furthermore, we conjecture that the D-family-matching problem 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.

Our algorithms should prove of paramount importance to identify stable meta-clusters amidst clusterings (from different algorithms, or from the same algorithm with different parameters). Formalizing the notion of stability for meta-clusters may indeed leverage clusterings by removing the arbitrariness inherent to the various algorithms and options available.

References

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

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

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

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

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

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

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

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

[9] 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.

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

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

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

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

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

[15] A.K. Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.

[16] Viggo Kann. Maximum bounded 3-dimensional matching is max snp-complete. *Inf. Process. Lett.*, 37(1):27–35, January 1991.

[17] R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad and spectral. *Journal of the ACM (JACM)*, 51(3):497–515, 2004.

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

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

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

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

[22] H. Nagamochi and T. Ibaraki. A fast algorithm for computing minimum 3-way and 4-way cuts. *Mathematical Programming*, 88(3):507–520, 2000.

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

[24] 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.

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

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

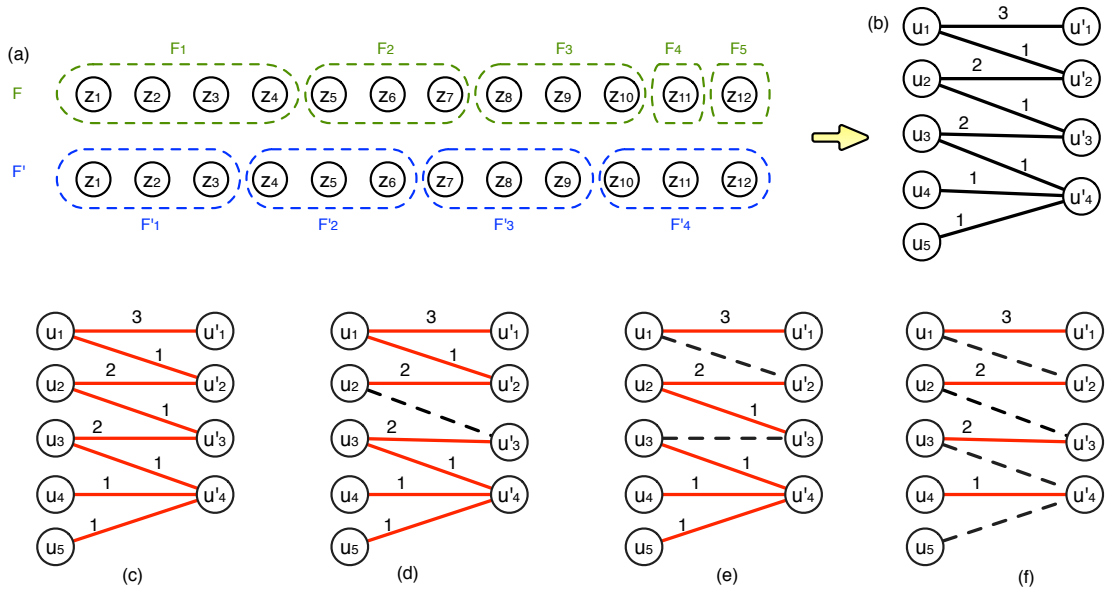
[27] R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005.

[28] L. Zhao, H. Nagamochi, and T. Ibaraki. *ISAAC*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

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

A Appendix - Detailed example

Figure 6 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 $\Phi(\mathcal{S}) = \Phi_2(G) = 9$. **(f)** Optimal solution $\mathcal{S}$ for $D = 1$ with $\Phi(\mathcal{S}) = \Phi_1(G) = 8$.



B Appendix - Table of 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
$N_G(v) = \{v' \mid \{v, v'\} \in E\}$	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

C Appendix - Equivalent definition of the D-family-matching problem

We first prove Lemma 1.

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. □

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

Definition. 6 (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 $i, j \in \{1, \ldots, k\}$, $i \neq j$, 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. 7 (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 4. *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 $\Phi(\mathcal{S}) \geq L$.*

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

D.1 Proof of Theorem 1

This section is devoted to the proof of Theorem 1. For the sake of readability, we splitted this proof into three parts: Theorems 5, 8 and 9. Notice that the last two proofs are quite similar.

D.1.1 Part I: $\Delta = 4$, fixed values of weights

Theorem. 5. *For any $D \geq 2$, the D-family-matching problem is NP-complete even if the maximum degree Δ is at most 4 and the weights are 2 and 5.*

In our reduction, we use a special case of set packing problem, a well known NP-complete problem [18]. 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 $k \geq 1$, **set packing problem** consists in determining whether there exists a packing $\mathcal{C}$ of size $|\mathcal{C}| = k$. Set packing problem is NP-complete even if $|Y_i| = 3$ for every $i \in \{1, \ldots, p\}$. Furthermore, if x_i is in at most 3 sets, for every $i \in \{1, \ldots, t\}$, then the problem is MAX SNP-complete [16]. For such a special case, the problem is known as 3-dimensional matching problem.

We initially prove the result for $D = 2$. 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 2-family-matching problem (Definition 8).

Definition. 8 (Construction of the intersection graph G for the 2-family-matching problem)**.** *The intersection graph $G = (U, U', E, w)$ is defined as follows.*

- *Set $U = U_1 \cup U_2$, where*
  - *$U_1 = \{u_1^1, \ldots, u_p^1\}$ corresponds to Y*
  - *and $U_2 = \{u_1^2, \ldots, u_p^2\}$.*
- *Set $U' = U'_1 \cup U'_2$, where*
  - *$U'_1 = \{u_1'^1, \ldots, u_t'^1\}$ corresponds to X*
  - *and $U'_2 = \{u_1'^2, \ldots, u_p'^2\}$.*
- *Set $E = E_a \cup E_b \cup E_c$, where*
  - *$E_a = \{\{u_i^2, u_i'^2\} \mid 1 \leq i \leq p\}$,*
  - *$E_b = \{\{u_i^1, u_i'^2\} \mid 1 \leq i \leq p\}$,*
  - *and $E_c = \{\{u_i^1, u_j'^1\} \mid x_j \in Y_i, 1 \leq i \leq p, 1 \leq j \leq t\}$.*
- *Set $w_e = 5$ for every $e \in E_a \cup E_b$ and $w_e = 2$ for every $e \in E_c$.*

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

Lemma. 7. *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 2-family-matching problem for G such that $\Phi(\mathcal{S}) \geq 10p + k$.*

Proof of Lemma 7. 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 2-family-matching problem for G such that $\Phi(\mathcal{S}) = 10p + k$. Assume that $\mathcal{C} = \{Y_1, \ldots, Y_k\}$ (we permute the indices otherwise). Let $\mathcal{S} = \{S_1, \ldots, S_p\}$, where $S_i = \{u_i^1\} \cup N_G(u_i^1)$ for every $i \in \{1, \ldots, k\}$ and $S_i = \{u_i^1, u_i'^2, u_i^2\}$ 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 2. Finally, we get

$$\Phi(\mathcal{S}) = \Phi(\{S_1, \ldots, S_k\}) + \Phi(\{S_{k+1}, \ldots, S_p\}) = 11k + 10(p - k) = 10p + k.$$

Thus, we have proved that $\mathcal{S}$ is a solution for the 2-family-matching problem for G such that $\Phi(\mathcal{S}) = 10p + k$. □

Lemma. 8. *If there is a solution $\mathcal{S}$ for the 2-family-matching problem for G such that $\Phi(\mathcal{S}) \geq 10p + k$, 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 8. Consider any optimal solution $\mathcal{S}$ for the 2-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 claim.

Claim 6. *Consider any node $u^1 \in U_1$. Let $S_1 \in \mathcal{S}$ be such that $u^1 \in S$. Then, $|U'_1 \cap S_1| \in \{0, 3\}$.*

Proof of Claim 6. By contradiction. Assume that there exists a node $u^1 \in U_1$ and a set $S_1 \in \mathcal{S}$ such that $u^1 \in S$ and $|U'_1 \cap S_1| \in \{1, 2\}$. Let $u'^2 \in U'_2$ be such that $u'^2 \in N_G(u^1)$ and let u^2 be such that $\{u'^2, u^2\} \in E$. There are two cases.

- First, assume that $u'^2 \in S_1$. Thus, for any $S' \in \mathcal{S}$, then $u^2 \notin S'$. In particular, $u^2 \notin S_1$ because otherwise $G[S_1]$ would have diameter at least three. We get that $|S_1| \in \{3, 4\}$ and $\sum_{e \in E(G[S_1])} w_e \in \{7, 9\}$. Without loss of generality, assume that $\mathcal{S} = \{S_1, \ldots, S_{p'}\}$ for some $p' \geq 1$. We construct $\mathcal{S}'$ from $\mathcal{S}$ as follows. Set $\mathcal{S}' = \{S'_1, S_2, \ldots, S_{p'}\}$, where $S'_1 = \{u'^2, u^2\}$. We get that $\Phi(\mathcal{S}') - \Phi(\mathcal{S}) = 1$ if $|U'_1 \cap S_1| = 2$ and $\Phi(\mathcal{S}') - \Phi(\mathcal{S}) = 3$ if $|U'_1 \cap S_1| = 1$. A contradiction because $\mathcal{S}$ is an optimal solution for the 2-family-matching problem.
- Second, assume that $u'^2 \notin S_1$. There are two sub-cases.
  - There exists $S_2 \in \mathcal{S}$ such that $\{u'^2, u^2\} \in E(G[S_2])$. We necessarily have $|S_2| = 2$ because $N_G(u^2) = \{u'^2\}$ and $N_G(u'^2) = \{u^2, u^1\}$. Without loss of generality, assume that $\mathcal{S} = \{S_1, \ldots, S_{p'}\}$ for some $p' \geq 1$. We construct $\mathcal{S}'$ from $\mathcal{S}$ as follows. Set $\mathcal{S}' = \{S'_2, S_3, \ldots, S_{p'}\}$, where $S'_2 = S_2 \cup \{u^1\} = \{u'^2, u^2, u^1\}$. Since $w_{u'^2,u^2} = w_{u'^2,u^1} = 5$, we get that $\Phi(\mathcal{S}') - \Phi(\mathcal{S}) = 1$ if $|U'_1 \cap S_1| = 2$ and $\Phi(\mathcal{S}') - \Phi(\mathcal{S}) = 3$ if $|U'_1 \cap S_1| = 1$. A contradiction because $\mathcal{S}$ is an optimal solution for the 2-family-matching problem.
  - For any $S' \in \mathcal{S}$, then $\{u'^2, u^2\} \notin E(G[S'])$. We have that both u'^2 and u^2 are not in a set of $\mathcal{S}$ because $N_G(u^2) = \{u'^2\}$ and $N_G(u'^2) = \{u^2, u^1\}$. Indeed, recall that every set of $\mathcal{S}$ contains at least two nodes. (If it is not the case, we can remove u'^2 and u^2 without decreasing the score.) Thus, assume that $\mathcal{S} = \{S_1, \ldots, S_{p'}\}$ for some $p' \geq 1$ and for every $i \in \{1, \ldots, p'\}$, then $u'^2 \notin S_i$ and $u^2 \notin S_i$. We construct $\mathcal{S}'$ from $\mathcal{S}$ as follows. Set $\mathcal{S}' = \{S'_1, S_2, \ldots, S_{p'}\}$, where $S'_1 = \{u'^2, u^2, u^1\}$. Again, we get that $\Phi(\mathcal{S}') - \Phi(\mathcal{S}) \in \{1, 3\}$. A contradiction because $\mathcal{S}$ is an optimal solution for the 2-family-matching problem.

Thus, we have proved that $|U'_1 \cap S_1| \in \{0, 3\}$. □

By Claim 6, we get that for every $i \in \{1, \ldots, p\}$, then there exists $S \in \mathcal{S}$ such that

- either $S = \{u'^2_i\} \cup N_G(u'^2_i) = \{u^2_i, u'^2_i, u^1_i\}$ and $\sum_{e \in E(G[S])} w_e = 10$
- or $S = \{u^1_i\} \cup N_G(u^1_i) = \{u'^2_i, u^1_i, u'^1_{j_1}, u'^1_{j_2}, u'^1_{j_3}\}$ and $\sum_{e \in E(G[S])} w_e = 11$, where $N_G(u^1_i) \cap U'_1 = \{u'^1_{j_1}, u'^1_{j_2}, u'^1_{j_3}\}$ for some $j_1, j_2, j_3 \in \{1, \ldots, t\}$, $j_1 < j_2 < j_3$.

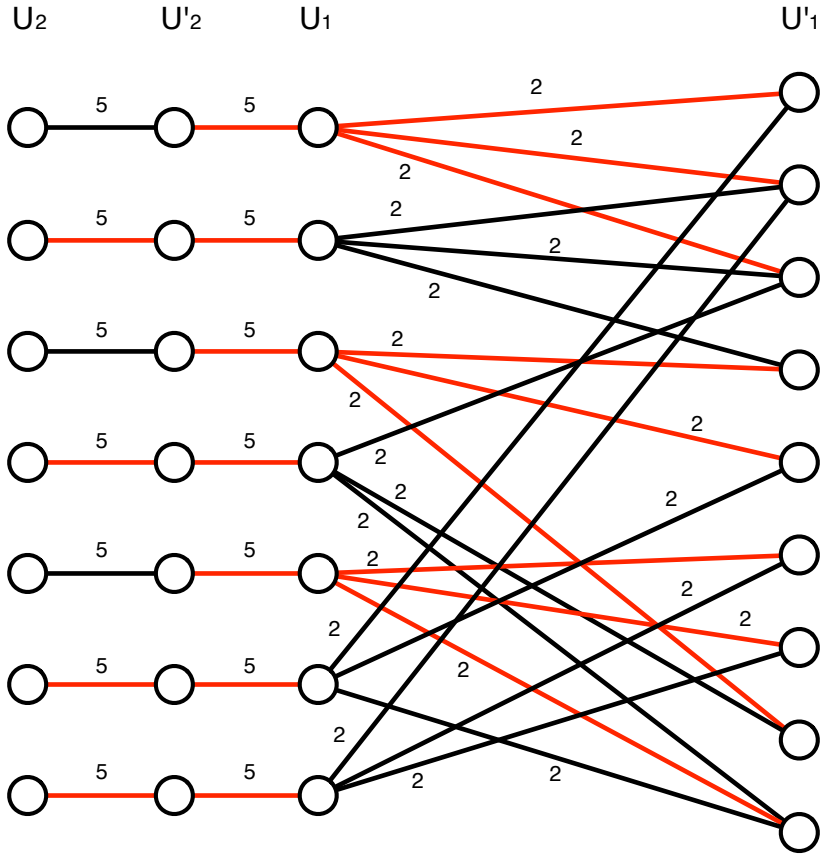
Observe that we cannot have two sets $S, S' \in \mathcal{S}$ such that $S = \{u^2_i, u'^2_i\}$ and $S' = \{u^1_i, u'^1_{j_1}, u'^1_{j_2}, u'^1_{j_3}\}$ because we have assumed that $\mathcal{S}$ has a minimum number of sets. Indeed, $\sum_{e \in E(G[S])} w_e + \sum_{e \in E(G[S'])} w_e = 11$ but it is possible to consider one single set with same score (second case described before).

We deduce the following claim.

Claim 7. *For any node $u'^1 \in U'_1$ and for any $S \in \mathcal{S}$, then $|N_G(u'^1) \cap S| \in \{0, 1\}$.*

By previous remarks and Claim 7, we get that $\mathcal{S}$ contains exactly p sets. Recall that we assume that every set contains at least two nodes. Let $\mathcal{S} = \{S_1, \ldots, S_p\}$. We get that $\sum_{e \in E(G[S_i])} w_e \in \{10, 11\}$ and $\Phi(\mathcal{S}) = 11k + 10(p - k)$ for some $k \in \{1, \ldots, p\}$. Thus, it means that there exist $i_1, \ldots, i_k \in \{1, \ldots, p\}$, $i_1 < \ldots < i_k$, such that, for every $i \in \{i_1, \ldots, i_k\}$, then $S_i = \{u^1_i\} \cup N_G(u^1_i)$ and $\sum_{e \in E(G[S_i])} w_e = 11$. We deduce that $N_G(u^1_i) \cap N_G(u^1_{i'}) = \emptyset$ for any $i, i' \in \{i_1, \ldots, i_k\}$, $i \neq i'$.

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



Thus, by construction of G, we finally obtain that $\mathcal{C} = \{Y_{i_1}, \ldots, Y_{i_k}\}$ is a set packing for the instance $\mathcal{I}_{sp}$ of size $|\mathcal{C}| = k$. □

We are now able to prove the theorem.

Proof of Theorem 5. First, the reduction (Definition 8) can be clearly done in polynomial time. Second, the decision version of the 2-family-matching is in NP because, given any $\mathcal{S}$, one can check in polynomial time if $\mathcal{S}$ is a 2-family-matching and one can compute in polynomial time the score $\Phi(\mathcal{S})$. Finally, Lemmas 7 and 8 prove that there is a solution $\mathcal{C}$ for the instance $\mathcal{I}_{sp}$ of set packing problem such that $|\mathcal{C}| \geq k$ if and only if there is a solution $\mathcal{S}$ for the 2-family-matching problem for G such that $\Phi(\mathcal{S}) \geq 10p + k$. Thus, the decision version of the 2-family-matching is NP-complete.

To conclude the proof, we show how to get the result for any $D \geq 3$. First, we have a path composed of D edges (each of weight 5) from every node of U_1, instead of a path of 2 edges (each of weight 5). Say otherwise, we remove U_2 and U'_2, and we add such a path from each node of U_1. The reduction can be done in polynomial time and the decision version of the D-family-matching is also clearly in NP. Furthermore, there is a solution $\mathcal{C}$ for the instance $\mathcal{I}_{sp}$ of set packing problem such that $|\mathcal{C}| \geq k$ if and only if there is a solution $\mathcal{S}$ for the D-family-matching problem for G such that $\Phi(\mathcal{S}) \geq 5Dp + k$. Thus, the decision version of the

D-family-matching is NP-complete. □

To illustrate the proof of Theorem 5, consider the instance $\mathcal{I}_{sp}$ of set packing problem, where $X = \{x_1, \ldots, x_9\}$, a family $Y = \{Y_1, \ldots, Y_7\}$ 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 $p = 7$. The graph G depicted in Figure 7 is the graph obtained from Definition 8. There is a set packing $\mathcal{C} = \{Y_1, Y_3, Y_5\}$ of size $k = 3$ and there is a 2-family-matching $\mathcal{S}$ such that $\Phi(\mathcal{S}) = 10p + k = 73$ (depicted in red in Figure 7).

D.1.2 Part II: D = 2, Δ = 3, unary weights

We prove the following:

Theorem. 8. *The 2-family-matching problem is NP-complete 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 NP-complete on graphs on maximum degree 3 [7].

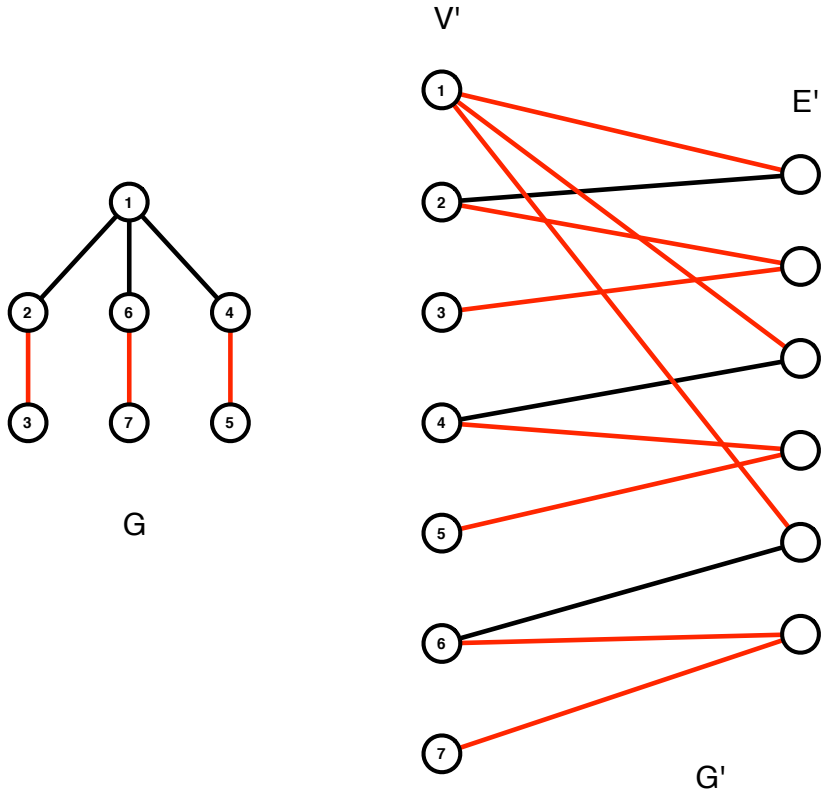
Let $G = (V, E)$ be a graph of maximum degree 3, with $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$. We construct a bipartite graph G' composed of the vertex set $V' \cup E'$, where $V' = \{v'_1, \ldots, v'_n\}$ and $E' = \{e'_1, \cdots e'_m\}$. Then, if $e_j = \{v_a, v_b\}$ is an edge of 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 $|M| \geq k$. □

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



To illustrate the proof of Theorem 8 consider the instance of Induced Matching problem depicted in Figure 8 (left). Figure 8 (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.

D.1.3 Part III: D > 2, Δ = 3

We prove the following result:

Theorem. 9. *For any $D \geq 3$, the D-family-matching problem is NP-complete 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.

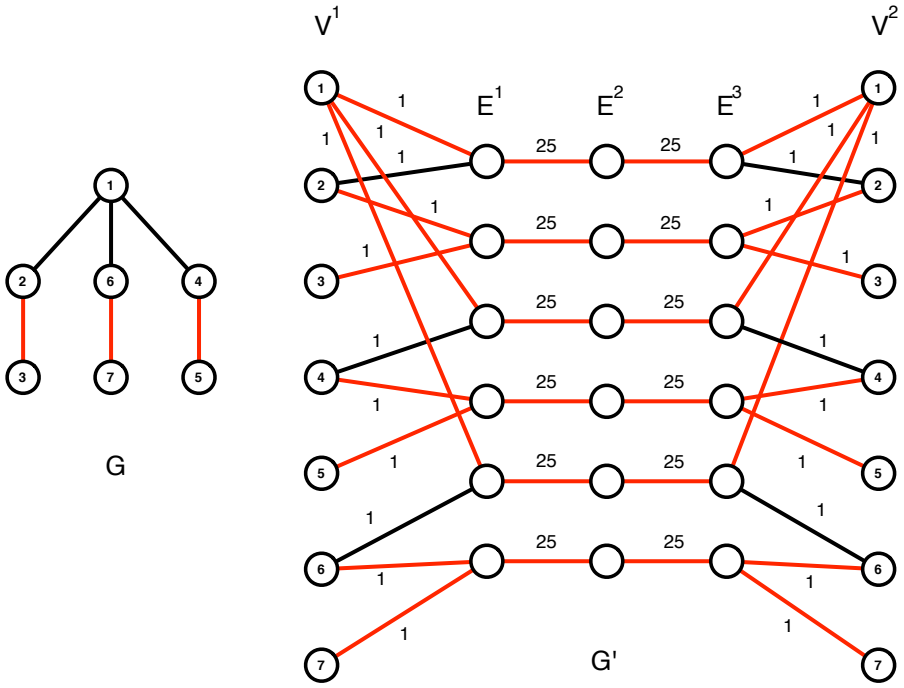
Applying this modification leads to a solution $\mathcal{S}$ such that M represents an induced matching in G. Finally, the number of edges contained in a cluster of $\mathcal{S}$ among those between V^1 and E^1 is at most $2|M| + |E \setminus M| = m + |M|$, and at least $m+k$, thus $|M| \geq k$. □

To illustrate the proof of Theorem 9, consider the instance of Induced Matching problem depicted in Figure 9 (left). Figure 9 (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 $18 + 12W = 318$.

D.2 Unbounded ratio between scores by increasing the diameter by one

Proof of Lemma 4. Let $t = n - 1$. Let $Z = \{z_1, \dots, z_t\}$, $F = \{F_1\}$, and $F' = \{F'_1, \dots, F'_t\}$ be an instance of the D-family-matching problem, where $F_1 = \{z_1, \dots, z_t\}$ and $F'_i = \{z_i\}$ for every $i \in \{1, \dots, t\}$. The intersection graph $G = (U, U', E, w)$ is such that $U = \{u_1\}$, $U' = \{u'_1, \dots, u'_t\}$, $E = \{\{u_1, u'_i\} \mid 1 \leq i \leq t\}$, and $w_e = 1$ for every $e \in E$. We first prove that any solution $\mathcal{S}^{D=1}$

Figure 9 Illustration of the proof of Theorem 9. See details in the text.



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 $\Phi(\mathcal{S}^{D=2}) = t = n - 1$. □

D.3 Optimizing first the score of a single set can be arbitrarily bad

Proof of Lemma 3. 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 3. □

E Appendix - Polynomial time dynamic programming algorithms for some classes

E.1 The D-family-matching problem for trees

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*

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

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

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 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 $j \in \{1, \ldots, q\}$, 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 ($j = 1, \ldots, q$), 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 $j' \in \{1, \ldots, q\}$, $j' \neq j$, 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 $\max_{i' \in \{1,\ldots,D\}} \Psi_D(v_{j'}, i')$.
- $S_v \cap V(T_{v_j}) \neq \emptyset$. In that case, the largest possible score is $\max_{i' \in \{1,\ldots,D-i-1\}} \Psi_D(v_{j'}, i') + w_{v,v_{j'}}$. 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 $D - i - 1$. Otherwise, the diameter of S_v would be at least $D + 1$.

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 $\Psi_D(v, i)$. □

Note that Theorem 2 is polynomial because $\Delta \leq n - 1$ and $D \leq n - 1$.

E.2 The D-family-matching problem for paths

In this section, we illustrate the result of Theorem 2 by considering paths. Consider an intersection path $G = (V, E, w)$. By Theorem 2, 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 9 a better time complexity algorithm for the D-family-matching problem. Indeed, the time complexity is $O(Dn)$.

Lemma. 9 (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 9. Let $V = \{v_1, \ldots, v_n\}$. Let $E = \{\{v_j, v_{j+1}\} \mid 1 \leq j \leq n-1\}$. We define the function Ψ_D as follows. For every $t \in \{1, \ldots, n\}$ and every $i \in \{\max(1, t - D), \ldots, t + 1\}$, 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 $\{v_1, \ldots, v_t\}$, such that $\{v_i, \ldots, v_t\} \in \mathcal{S}$. 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 $t \in \{1, \ldots, n-1\}$. Suppose we have computed $\Psi_D(v_{t'}, i)$ for every $t' \in \{1, \ldots, t\}$ and every $i \in \{\max(1, t - D), \ldots, t' + 1\}$. 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 9, 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. □

E.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

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

To conclude Section E, we address in Corollary 2 the results of Theorem 3 in terms of the original problem (that is for the equivalent definition proved in Section C).

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

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

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 $F \cup F'$.

F Appendix - Generic approach based on spanning trees

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.

F.1 Dynamic programming algorithms under spanning tree constraint

Proof of Lemma 6. 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 $\Psi_{k',D}(v, H) = -\infty$.

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 6, 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 $\mathcal{S}$). □

F.2 Algorithms based on spanning trees

Proof of Lemma 5. For some $k \geq 1$, consider an optimal solution $\mathcal{S} = \{S_1, \ldots, S_k\}$ for the D-family-matching problem for G. For every $i \in \{1, \ldots, k\}$, 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 $i \in \{1, \ldots, k\}$. By construction of T, $\mathcal{S}$ is an admissible solution for the D-family-matching problem for G constrained by T. Thus, $\Phi_D(G, T) = \Phi_D(G)$. □

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:*

- $\Pi(\mathcal{M}) \Leftrightarrow |\mathcal{M}| = |\mathcal{T}(G)|$,
- $\mathcal{R}(G, t) = T^t$, *where* $\mathcal{T}(G) = \{T^1, \ldots, T^{|\mathcal{T}(G)|}\}$,
- *and Algorithm* $\mathcal{A}(G, T^t, D)$ *returns* $\Phi_D(G, T^t)$ *(Lemma 6).*

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. □

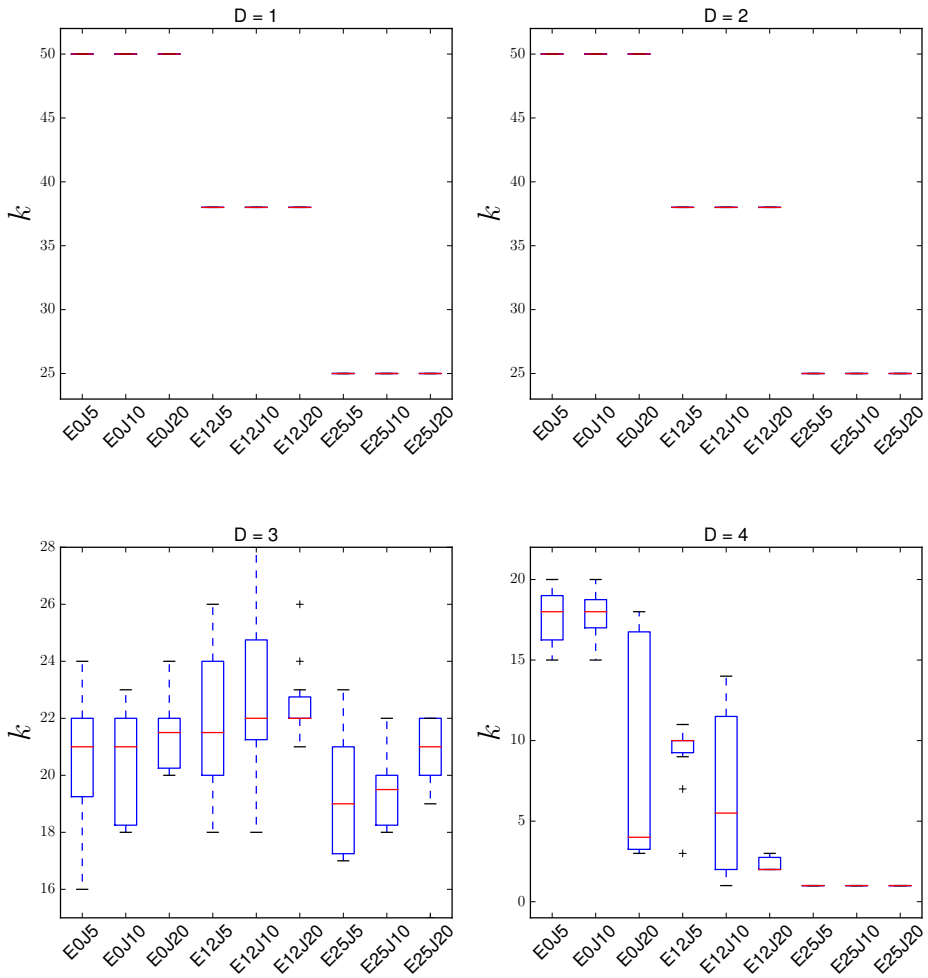
Corollary. 4. *Given any intersection graph G, Algorithm 1 returns a 2Δ-approximation for the 2-family-matching problem for G if:*

- $\Pi(\mathcal{M}) \Leftrightarrow |\mathcal{M}| = |\mathcal{T}(G)|$,
- $\mathcal{R}(G,t) = T^t$, *where* $\mathcal{T}(G) = \{T^1, \ldots, T^{|\mathcal{T}(G)|}\}$,
- *and Algorithm* $\mathcal{A}(G, T^t, D)$ *returns* $\Phi_D(T^t)$ *(Theorem 2).*

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

G Appendix - Experiments

Figure 10 Solution of our algorithm $STS(G, D)$ with $(t = 3\ 000, r = 50)$. We generate a solution for $n_i = 10\ 000$ random spanning trees as well as for the maximum spanning tree. We keep the solution with the highest Φ. Number k as a function of the 9 scenarii.



Results

Figure 11 Solution of our algorithm $STS(G, D)$ with $(t = 3\ 000, r = 50)$. We generate a solution for $n_i = 10\ 000$ random spanning trees as well as for the maximum spanning tree. We keep the solution with the highest Φ. Score Φ as a function of the 9 scenarii.

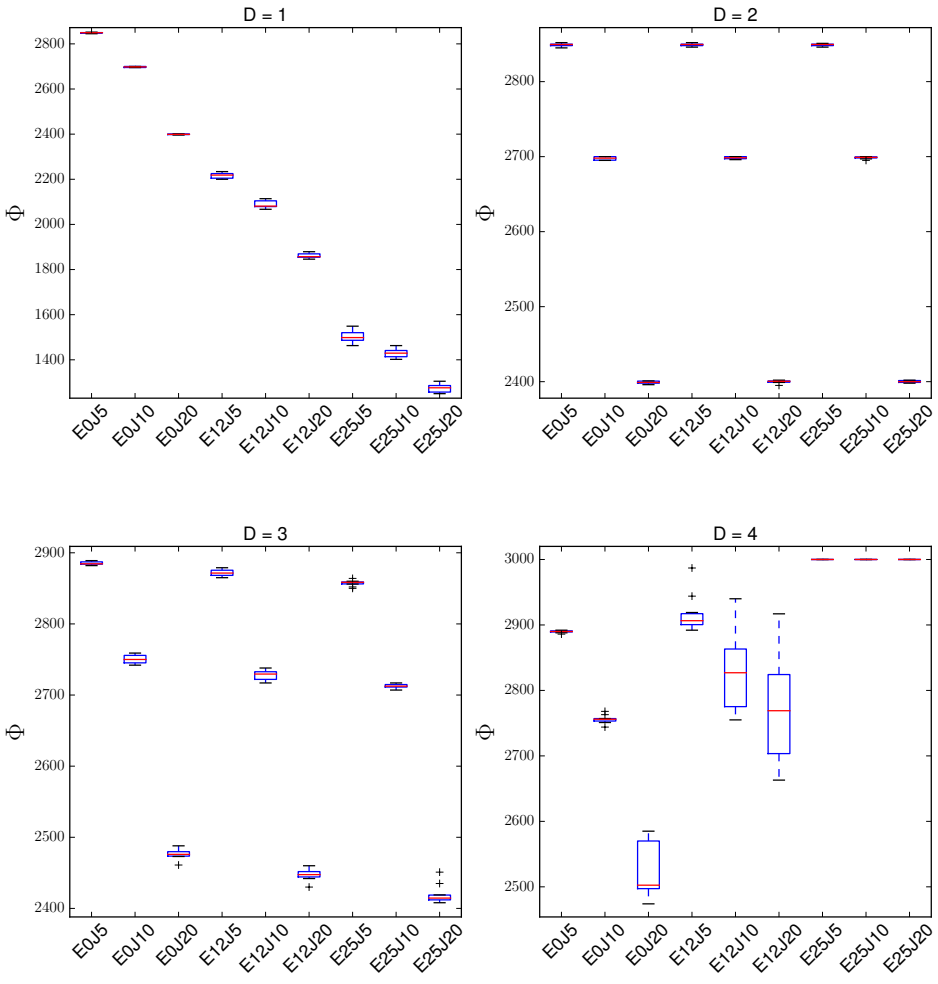
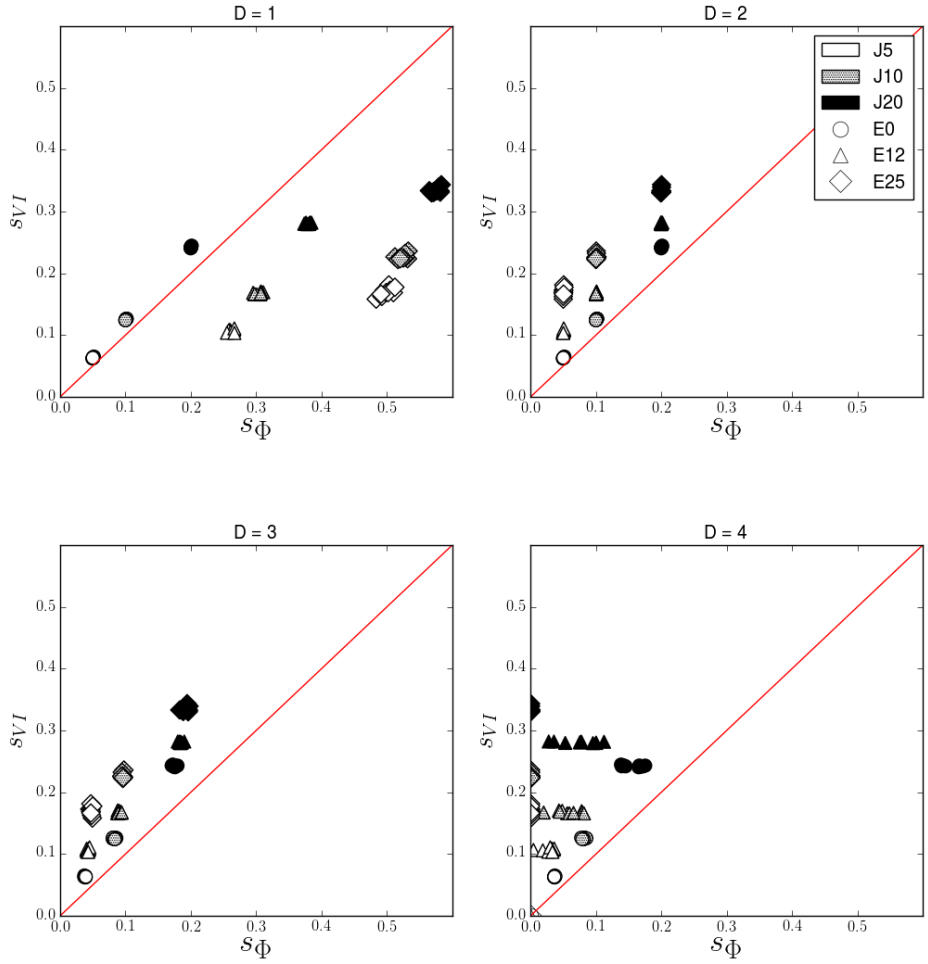
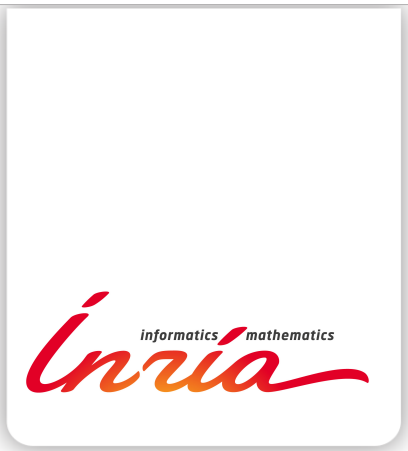


Figure 12 Algorithm $STS(G, D)$ compared to VI for clusterings with $(t = 3\ 000, r = 50)$. Both scores were normalized to be contained in the $[0, 1]$ interval.





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