HAL
open science

Clustering stability revealed by matchings between clusters of clusters

Frédéric Cazals, Dorian Mazauric, Romain Tetley

To cite this version:

Frédéric Cazals, Dorian Mazauric, Romain Tetley. Clustering stability revealed by matchings between clusters of clusters. [Research Report] RR-8992, Inria Sophia Antipolis; Université Côte d'Azur. 2016, pp.51. hal-01410396

HAL Id: hal-01410396

https://inria.hal.science/hal-01410396v1

Submitted on 6 Dec 2016

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

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

Clustering stability revealed by matchings between clusters of clusters

Frédéric Cazals and Dorian Mazauric and Romain Tetley

RESEARCH REPORT
N° 8992
Decembre 2016

Project-Team Algorithms-Biology-Structure

ISSN 0249-6399 ISRN INRIA/RR--8992--FR+ENG

Clustering stability revealed by matchings between clusters of clusters

Frédéric Cazals and Dorian Mazauric and Romain Tetley

Project-Team Algorithms-Biology-Structure

Research Report n° 8992 — Decembre 2016 — 51 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 (k, D) and D-family-matching problems on intersection graphs, with k the number of meta-clusters and D the upper-bound on the diameter of the graph induced by the clusters of any meta-cluster. First we prove hardness and inapproximability results. Second, we design exact polynomial time dynamic programming algorithms for some classes of graphs (in particular trees). Then, we prove efficient (exact, approximation, and heuristic) algorithms, based on spanning trees, for general graphs.

Practically, 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, we show how our methods can be used to identify notorious instabilities of the k-means algorithm.

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

La stabilité de clusterings révélée par des correspondances entre 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 les (k, D) et D-family-matching problèmes à partir du graphe intersection, avec k le nombre de meta-clusters et D 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 utilisons cette faculté pour identifier des instabilités notoires de l'algorithme k-means.

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

Contents

1 Introduction **5**
1.1 Clusterings: generation, comparison and stability assessment 5
1.2 Main contributions 6

2 Comparison of clusterings: formalization as graph problems **7**
2.1 The (k,D)-family-matching problem 7
2.2 The D-family-matching problem 9
2.3 Polynomial time algorithm when D = 1 9

3 The family-matching problems are very hard to solve **10**
3.1 NP-completeness results 10
3.2 Hardness of approximation for general graph even with k = D = 1 10
3.3 Unbounded ratio between scores by increasing the diameter by one 10
3.4 Optimizing first the score of a single set can be arbitrarily bad 11

4 Polynomial time dynamic programming algorithms for some classes **11**

5 Efficient algorithms for general graphs **13**
5.1 Dynamic programming algorithms under spanning tree constraint 13
5.2 Algorithms based on spanning trees 14
5.2.1 Generic algorithms 14
5.2.2 Exact algorithms 15
5.2.3 Approximation and heuristic algorithms 15

6 Experiments **15**
6.1 Implementation 15
6.2 Experiments on random clusterings 16
6.3 Application to the instability of k-means 17

7 Outlook **17**

A Appendix - Table of notations **21**

B Appendix - Equivalent definitions of the family-matching problems **21**

C Appendix - The family-matching problems are very hard to solve **22**
C.1 NP-completeness results 22
C.1.1 NP-completeness of the (k,D)-family-matching problem 23
C.1.2 NP-completeness of the 2-family-matching problem 25
C.2 Hardness of approximation for general graph even with k = D = 1 29
C.3 Unbounded ratio between scores by increasing the diameter by one 29
C.4 Optimizing first the score of a single set can be arbitrarily bad 29

D Appendix - Polynomial time dynamic programming algorithms for some classes **30**
D.1 The (k,D)-family-matching problem for trees 30
D.2 The D-family-matching problem for trees 32
D.3 The (k,D)-family-matching problem for paths 33
D.4 The D-family-matching problem for paths 35
D.5 The (k,D)-family-matching problem for cycles 36

D.6 The D-family-matching problem for cycles 36
D.7 The (k,D)-family-matching problem for union of graphs 37
D.8 The D-family-matching problem for union of graphs 38

E Appendix - Efficient algorithms for general graphs 39
E.1 Warm up 39
E.2 Dynamic programming algorithms under spanning tree constraint 40
E.3 Algorithms based on spanning trees 45

F Appendix - Experiments 46

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 [28, 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 [9]. *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 [24]. Finally, *spectral clustering* methods define clusters from the top singular vectors of the matrix representing the data (or their similarity) [27]. From an axiomatic standpoint, this diversity is related to the impossibility theorem for clustering [18], which stipulates that no clustering scheme satisfying three simple properties (*scale-invariance, richness, consistency*) exists. Nevertheless, such an 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 [16]. 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, 7], 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

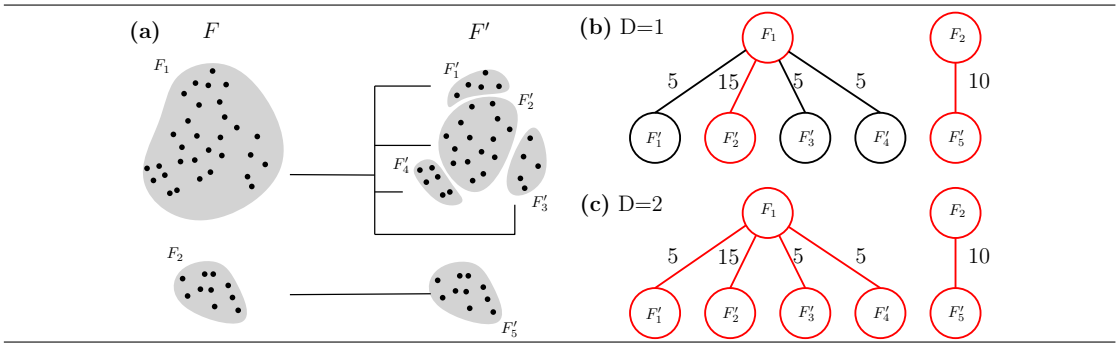
$$d_{MMD}(F, F') = \frac{1}{t} \min_{\pi \in \Pi} \sum_{i=1}^{t} \mathbf{1}_{F(x_i) \neq \pi(F'(x_i))}, \tag{1}$$

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 also 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.

In *information theoretical methods* [22], the coherence between clusters of F and F' is assessed using the variation of information between the clusterings. In short, VI is defined from the mutual

Figure 1 Comparing two clusterings of the same 2D data set involving 40 points. 30 and 10 points. Clustering F' contains 5 clusters of respectively 5, 15, 5, 5, and 10 points. Each edge displays the number of points shared by 2 clusters (Definition 1). Methods based on (maximum) graph matching would match F_1 with F'_2 and F_2 with F'_5. **(b)** Our method is parameterized by the diameter D of the sub-graphs connecting clusters within meta-clusters (in red). With $D = 1$, a matching is obtained. **(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 $\{F'_5\}$.



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* [30] 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 [25]–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 (Figure 1). We fill this gap by studying the problem of grouping clusters into meta-clusters, while defining a mapping between 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 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 [12, 23, 26, 29]. 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, shedding light on the way clusters from one have been merged / split / edited to define one clustering from the other, consists of the following contributions. In Section 2, we introduce two combinatorial optimization problems on the intersection graph, namely the (k, D)-family-matching problem and the D-family-matching problem, so as to compare two clusterings. In Section 3, we prove that the problems are very hard to solve: NP-completeness results, hardness of approximation, 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, unions of trees, graphs of maximum degree two). In Section 5, we describe efficient (exact, approximation, and heuristic) 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, we show how our methods can be used to identify notorious instabilities of the k-means algorithm [27].

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

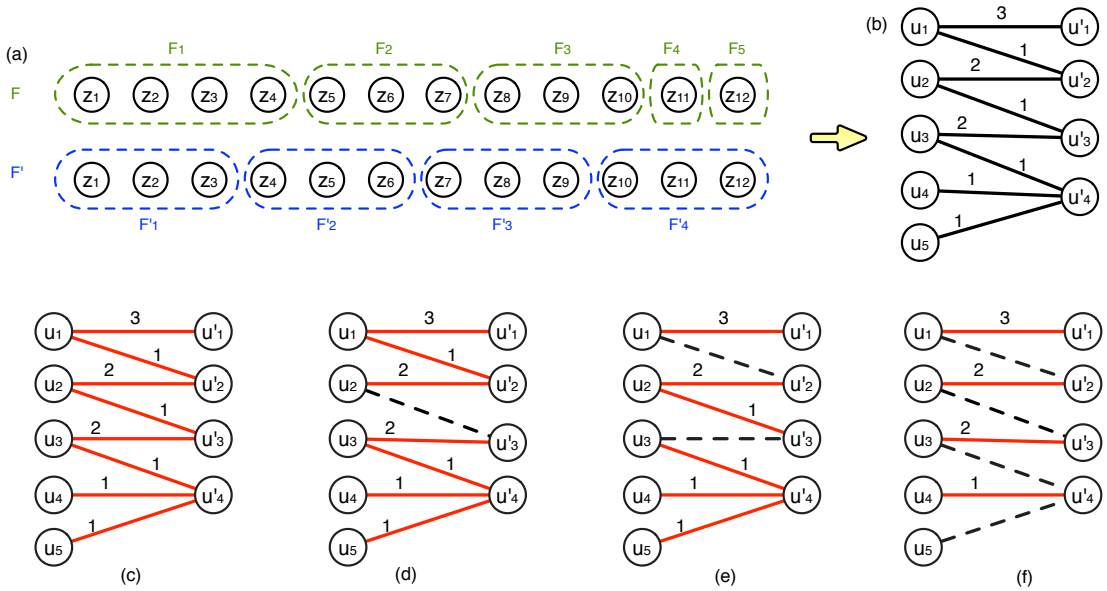
2 Comparison of clusterings: formalization as graph problems

2.1 The (k,D)-family-matching problem

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 $i, j \in \{1, \ldots, r'\}$, $i \neq j$. Figure 2 (a) describes two simple families F and F' with $t = 12$, $r = 5$, and $r' = 4$. Let us now define the notion of edge-weighted intersection graph.

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 $w_e = |F_i \cap F'_j|$.*

Figure 2 Simple instance of the (k,D)-family-matching problem and solutions. (a) Simple instance of the (k, 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 $k = 1$ and $D \geq 7$ with $\Phi(\mathcal{S}) = \Phi_{1,D}(G) = 12$. **(d)** Optimal solution $\mathcal{S}$ for $k = 2$ and $D = 3$ with $\Phi(\mathcal{S}) = \Phi_{2,3}(G) = 11$. **(e)** Optimal solution $\mathcal{S}$ for $k = 3$ and $D = 2$ with $\Phi(\mathcal{S}) = \Phi_{3,2}(G) = 9$. **(f)** Optimal solution $\mathcal{S}$ for $k = 4$ and $D = 1$ with $\Phi(\mathcal{S}) = \Phi_{4,1}(G) = 8$.



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 $\{u, u'\} \in E$. Figure 2 (b) describes an intersection graph corresponding to the two families F and F' depicted in Figure 2 (a). 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 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 (k, D)-family-matching.

Definition. 2 ((k,D)-family-matching)**.** *Let $k, D \geq 1$ be two positive integers. Let $G = (V, E, w)$ be an intersection graph. A (k, D)-family-matching for G is a family $\mathcal{S} = \{S_1, \ldots, S_k\}$ 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 (k, D)-family-matching $\mathcal{S}$ is defined as follows:

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

Let $\mathcal{S}_{k,D}(G)$ be the set of all (k, D)-family-matching for G. We now formalize the (k, D)-family-matching problem.

Definition. 3 ((k,D)-family-matching problem)**.** *Let $k, D \geq 1$ be two positive integers. Given an intersection graph G, the (k, D)-family-matching problem consists in computing*

$$\Phi_{k,D}(G) = \max_{\mathcal{S} \in \mathcal{S}_{k,D}(G)} \Phi(\mathcal{S}).$$

Intuitively, we wish to compute a (k, D)-family-matching which minimizes the inconsistencies. Figure 2 describes a simple instance of the (k, D)-family-matching problem ($t = 12$, $r = 5$, $r' = 4$, and so $n = 9$). Figures 2 (c,d,e,f) represent optimal solutions for different values of k and D. In appendix (Section B), we prove an equivalent definition of the (k, D)-family-matching problem without intersection graph notion.

Remark 1.
- *It directly works with the items clustered rather than cluster representatives. This stresses the structure of clusters in terms of their constituting items.*
- *The clusterings compared may not be partitions of the input set of elements, which allows in particular discarding outliers.*

2.2 The D-family-matching problem

We define a variant of the (k, D)-family-matching problem: the D-family-matching problem.

Definition. 4 (D-family-matching problem)**.** *Let $D \geq 1$ be a positive integer. Given an intersection graph G, the D-family-matching problem consists in finding an optimal number of sets k such that the score of an optimal solution for the (k, D)-family-matching problem is minimum. Formally, the D-family-matching problem consists in computing*

$$\Phi_D(G) = \max_{k \in \{0,\ldots,n\}} \Phi_{k,D}(G).$$

The D-family-matching problem is motivated by the following conjecture: for any $D \geq 1$ and for any intersection graph $G = (V, E, w)$, then there exists $k \in \{0, \ldots, n\}$ such that

- $\Phi_{k_1,D}(G) \leq \Phi_{k_2,D}(G)$ for any $k_1, k_2 \in \{0, \ldots, k\}$, $k_1 \leq k_2$,
- and $\Phi_{k'_1,D}(G) \geq \Phi_{k'_2,D}(G)$ for any $k'_1, k'_2 \in \{k, \ldots, n\}$, $k'_1 \leq k'_2$.

2.3 Polynomial time algorithm when D = 1

Observe that the $(k, 1)$-family-matching problem and the 1-family-matching problem are equivalent to the problem of computing a maximum weighted matching in weighted bipartite graphs. Indeed, since $D = 1$, any sub-graph of diameter at most 1 is a complete sub-graph. Recall that any intersection graph G is a bipartite graph with positive integer weights (Definition 1). Thus, a complete sub-graph of G is either a single node or an edge (there is no triangle in a bipartite graph). Since the problem of computing a maximum weighted matching can be solved in $O(n^2 \log n + nm)$ [11], we deduce the following result.

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

3 The family-matching problems are very hard to solve

In this section, we prove that the family-matching problems are NP-complete, hard to approximate for general graphs, and that simple strategies can be arbitrarily bad. All the proofs can be found in appendix (Section C).

3.1 NP-completeness results

We prove in Theorems 1 and 2 that the decision versions of the (k, D)-family-matching problem and the D-family-matching problem are NP-complete for any $D \geq 2$ and $D = 2$, respectively. In our reduction, we use set packing problem, a well known NP-complete problem [17].

Theorem. 1. *Let $k \geq 1$ be a positive integer. Given any constant integer $D \geq 2$, the decision version of the (k, D)-family-matching problem is NP-complete.*

Theorem. 2. *The decision version of the 2-family-matching problem is NP-complete.*

An interesting open question is to know if there is a polynomial time algorithm with constant approximation ratio for the (k, D)-family-matching problem and/or for the D-family-matching problem (say otherwise, we do not know if these problems are in APX).

3.2 Hardness of approximation for general graph even with k = D = 1

In this section, we investigate the (k, D)-family-matching problem on general graphs (instead of bipartite graphs). Recall that a graph G is an intersection graph if G is bipartite and the weights are positive integers. But we think that it is important to exhibit the difficulty of the general problem, that is for general graphs. That allows us to show the intrinsic difficulty of the (k, D)-family-matching problem even for $k = D = 1$. If $w_e = 1$ for every $e \in E$, then the problem is equivalent to Clique problem (that consists in finding a largest complete sub-graph) that is very hard to approximate [14]. We prove in Lemma 3 that the problem is hard to approximate even for this very simple case.

Lemma. 3. *Let $\varepsilon > 0$. Unless $P = NP$, there is no polynomial time algorithm that approximates the $(1, 1)$-family-matching problem to within a factor better than $O(n^{1-\varepsilon})$ for any edge-weighted graph $G = (V, E, w)$ even if $w_e = 1$ for every $e \in E$.*

3.3 Unbounded ratio between scores by increasing the diameter by one

We analyze the ratio between the score of an optimal solution for the (k, D)-family-matching problem and the score of an optimal solution for the (k, D')-family-matching problem by only changing the diameter (the graph G and the number of sets k do not change). In other words, we analyze $\Phi_{k,D}(G)/\Phi_{k,D'}(G)$. If such a ratio is bounded for some classes of instances, then we can design an incremental algorithm for which the approximation will be a function of this ratio. Unfortunately, we prove in Lemma 4 that this ratio is not bounded even for very simple classes of instances.

Lemma. 4. *Let $k \geq 1$ be a positive integer. For any integer $n \geq 1$, then there exist an intersection graph $G = (V, E, w)$ composed of n nodes such that $\Phi_{k,2}(G)/\Phi_{k,1}(G) \geq n - 1$.*

3.4 Optimizing first the score of a single set can be arbitrarily bad

In order to solve the (k, D)-family-matching problem for G, suppose we have first computed an optimal solution $\mathcal{S}^{k=1}$ for the $(1, D)$-family-matching problem for G. In that case, such a solution forms a single sub-graph. We then enforce $\mathcal{S}^{k=1}$ to be a set of any (k, D)-family-matching. In other words, we assume that $\mathcal{S} = \{\mathcal{S}^{k=1}, S_1, \ldots, S_{k-1}\}$. In Lemma 5, we prove that there exist instances for which any algorithm that initially computes such a best single set $\mathcal{S}^{k=1}$, returns a solution that is arbitrarily far from the optimal solution for the (k, D)-family-matching problem, even for $D = 2$. In other words, computing first a largest weighted sub-graph with diameter at most D can be arbitrarily bad.

Lemma. 5. *For any integer $\lambda \geq 1$, then there exists an intersection graph $G = (V, E, w)$ such that $\Phi_{k,D}(G)/\Phi(\mathcal{S}^{max}) \geq \lambda - 2$ with $n = \lambda(\lambda - 1) + 1$, $k = \lambda$, $D = 2$, where $\mathcal{S}^{max}$ is an optimal solution for the (k, D)-family-matching problem for G with the constraint that $\mathcal{S}^{k=1} \in \mathcal{S}^{max}$, where $\mathcal{S}^{k=1}$ is such that $\Phi(\mathcal{S}^{k=1}) = \Phi_{1,D}(G)$.*

4 Polynomial time dynamic programming algorithms for some classes

In this section, we prove efficient exact dynamic programming algorithms for the (k, D)-family-matching problem and for the D-family-matching problem for some classes of graphs: trees, paths, cycles, unions of trees, graphs of maximum degree two. Most of them have polynomial-time complexity. Table 1 summarizes our results. All the proofs can be found in appendix (Section D). We first explain why we prove specific results for paths although a path is a tree of maximum degree two. By Lemma 6, given $D \geq 1$, there is an $O(D^2 n)$-time complexity algorithm for the D-family-matching problem when G is a path because $\Delta = 2$. However, we prove in Lemma 15 a better time complexity algorithm for the D-family-matching problem. Indeed, the time complexity is $O(Dn)$. 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.

Lemma. 6 (Computation of $\Phi_D(G)$ for trees)**.** *Let $D \geq 1$ be a positive integer. 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 $\Psi_D(T_v, i)$.

Table 1 Our exact dynamic programming algorithms for the (k, D)-family-matching problem and for the D-family-matching problem. For union of graphs, every maximal connected component can be solved in $O(C)$-time. Note the lesser complexity for the D-family-matching problem – versus (k, D)-family-matching problem, as complete dynamic programming tables are not maintained.

Problem	Class of graphs G	Time complexity	Reference
$\Phi_{k,D}(G)$	(Union of) Trees	$O(2^{\Delta}k^{\Delta}D^2\Delta^2 n)$	Lemmas 13, 17, Corollary 4
	(Union of) Trees with $\Delta = O(1)$	polynomial	Corollary 2
	(Union of) Paths	$O(kDn)$	Lemmas 14, 17, Corollary 4
	(Union of) Cycles and Graph of maximum degree two	$O(kD^2n)$	Lemmas 16, 17, Corollary 4
	Union of graphs	$O(\|cc(G)\|C)$	Lemma 17
$\Phi_D(G)$	(Union of) Trees	$O(D^2\Delta^2 n)$	Lemmas 6, 18, Corollary 6
	(Union of) Paths	$O(Dn)$	Lemmas 15, 18, Corollary 6
	(Union of) Cycles Graph of maximum degree two	$O(D^2n)$	Lemma 18, Corollaries 3, 6
	Union of graphs	$O(\|cc(G)\|C)$	Lemma 18

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,

$$\Phi(\mathcal{S}) = \Phi_D(G) = \max_{i \in \{-1,\ldots,D\}} \Psi_D(r, i).$$

5 Efficient algorithms for general graphs

In this section, we design algorithms for both versions of the problems and for general instances. Table 2 summarizes our results. All the proofs can be found in appendix (Section E). The first four algorithms consists in enumerating all the possible sets of k disjoint sub-graphs of diameter at most D. In next sections, we develop dynamic programming algorithms for a variant of the problem (Section 5.1) in order to design original (exact, approximation, and heuristic) algorithms for the (k, D)-FM and the D-family-matching problems (Section 5.2).

Let us introduce some notations. We define $N(\Delta, D)$ as the maximum number of nodes of a (Δ,D)-graph. A (Δ,D)-graph is a graph with maximum degree Δ and diameter at most D. 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 $h(G, T_r) = \max_{v\in V} h(G, T_r, v)$.

In the following, we focus on the D-family-matching problem. The results for the (k, D)-family-matching problem are similar and the proofs can be found in appendix (Section E).

Table 2 Our exact algorithms for the (k, D)-family-matching problem and for the D-family-matching problem for general graphs.

Problem	Class of graphs G	Time complexity	Reference
$\Phi_{k,D}(G)$	Any graph G	$O(n^k h(G)^k)$	Lemma 19
		$O(n^k 2^{k\Delta(1-\Delta^D)(1-\Delta)^{-1}})$	Lemma 20
		$O(n^{k.N(\Delta,D)})$	Lemma 21
		$O(n^{k(\Delta(\Delta-1)^D-2)(\Delta-2)^{-1}})$	Corollary 10
$\Phi_{k,D}(G, T_r)$	Any graph G	$O(k^\Delta h(G, T_r)^\Delta n)$	Lemma 22
	Any graph G such that $D, \Delta = O(1)$	polynomial	Corollary 11
	Any graph G such that $\Delta, h(G, T) = O(1)$		Corollary 12
$\Phi_D(G, T_r)$	Any graph G	$O(h(G, T_r)^\Delta n)$	Lemma 7
	Any graph G such that $D, \Delta = O(1)$	polynomial	Corollary 11
	Any graph G such that $\Delta, h(G, T) = O(1)$		Corollary 12
$\Phi_{k,D}(G)$	Any graph G	$O(\lvert\mathcal{T}(G)\rvert k^\Delta \max_{T_r\in\mathcal{T}(G)} h(G, T_r)^\Delta n)$	Lemma 23 Corollary 13
$\Phi_D(G)$		$O(\lvert\mathcal{T}(G)\rvert \max_{T_r\in\mathcal{T}(G)} h(G, T_r)^\Delta n)$	Lemma 8 Corollary 1

5.1 Dynamic programming algorithms under spanning tree constraint

We first define the variant of the D-family-matching problem.

Definition. 5 (D-family-matching constrained by a tree)**.** *Let $D \geq 1$ be a positive integer. Let $G = (V, E, w)$ be an intersection graph and let T_r be a spanning tree of G rooted at $r \in V$. A*

D-family-matching for G constrained by T_r is a D-family-matching $\mathcal{S}$ for G such that for every $S \in \mathcal{S}$, then there exists $H \in H(G, T_r)$ such that $S = H$.

The set $\mathcal{S}_D(G, T_r)$ is the set of all the D-family-matching constrained by T_r.

Definition. 6 (D-family-matching problem constrained by a tree)**.** *Let $D \geq 1$ be a positive integer. Given an intersection graph G and a rooted spanning tree T_r of G, the D-family-matching problem consists in computing*

$$\Phi_D(G, T_r) = \max_{\mathcal{S} \in \mathcal{S}_D(G, T_r)} \Phi(\mathcal{S}).$$

Lemma 7 proves a dynamic programming algorithm for this variant of the problem.

Lemma. 7 (Computation of $\Phi_D(G, T_r)$)**.** *Let $D \geq 1$ be a positive integer. Let $G = (V, E, w)$ be any intersection graph and let T_r be any rooted spanning tree of G. Then, there exists an $O(h(G, T_r)^{\Delta} n)$-time complexity algorithm for the D-family-matching problem for G constrained by T_r.*

5.2 Algorithms based on spanning trees

In this section, we design algorithms based on previous results for trees (constraints).

5.2.1 Generic algorithms

Our generic algorithm, described in Algorithm 1, consists in considering rooted spanning trees of G (in a sequential manner) and computing D-family-matching for G by using our efficient algorithms based on trees (e.g. $\Phi_D(T)$ or $\Phi_D(G, T)$). We now describe the main ingredients of Algorithm 1 by explaining the three parameters needed.

- **A property $\Pi(\mathcal{M})$.** While a given property Π, on the set $\mathcal{M}$ of computed D-family-matchings for G, is not satisfied, then we generate another rooted spanning tree T^t of G (by using $\mathcal{R}$) and compute a D-family-matching $\mathcal{S}^t$ for G based on T^t (by using $\mathcal{A}$).
- **A spanning tree generator $\mathcal{R}(G, t)$.** This function computes the rooted spanning tree T^t of G that is used at step $t \geq 1$ by Algorithm $\mathcal{A}$.
- **An algorithm $\mathcal{A}(G, T^t, D)$.** This algorithm computes a D-family-matching $\mathcal{S}^t$ for G based on the spanning tree $\mathcal{R}(G, t) = T^t$ of G.

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

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

```
1: M := ∅, t := 0
2: while ¬ Π(M) do
3:     t := t + 1
4:     Compute the spanning tree T^t := R(G, t)
5:     Compute S^t by using Algorithm A(G, T^t, D)
6:     M := M ∪ S^t
7: return S ∈ M of maximum score
```

5.2.2 Exact algorithms

We first prove in Lemma 8 that there exists at least one rooted spanning tree T of G such that the dynamic programming algorithm designed in Lemma 7 for the D-family-matching problem for G constrained by T, returns an optimal solution for the original D-family-matching problem for G.

Lemma. 8. *Let $D \geq 1$ be a positive integer. Let G be any intersection graph. Then, there exists a rooted spanning tree T of G such that $\Phi_D(G) = \Phi_D(G, T)$.*

Let $\mathcal{T}(G)$ be the set of all different rooted spanning trees of G. We deduce in Corollary 1 an exact algorithm for the D-family-matching problem for G.

Corollary. 1. *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 7).*

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

5.2.3 Approximation and heuristic algorithms

We design approximation and heuristic algorithms by using the following functions.

- $\Pi(\mathcal{M}) \Leftrightarrow |\Phi(\mathcal{S}) - \Phi(\mathcal{S}')| \leq \varepsilon$ for any $\mathcal{S}$, $\mathcal{S}' \in \mathcal{M}^p$, where $\mathcal{M}^p \subseteq \mathcal{M}$, $|\mathcal{M}^p| = p$, is a subset composed of D-family-matchings of largest scores, $p \geq 1$ and $\varepsilon > 0$ are given parameters.
- $\mathcal{R}(G, t)$ is a random rooted spanning tree T^t of G. The random function can be uniform or the probability can be function of the weight of the rooted spanning tree.
- Algorithm $\mathcal{A}(G, T^t, D)$ returns a D-family-matching $\mathcal{S}' = \{V(G[S_1]), \ldots, V(G[S_k])\}$ for G, where $\mathcal{S} = \{S_1, \ldots, S_k\}$ is a D-family-matching $\mathcal{S}$ for T^t such that $\Phi(\mathcal{S}) = \Phi_D(T^t)$ (Lemma 7). We return $\mathcal{S}'$ instead of $\mathcal{S}$ because we must add all the edges for which the two extremities are in a same set S_i.

6 Experiments

6.1 Implementation

We implemented two versions of Algorithm 1, parameterized by a graph $G = (V, E, w)$ and a diameter D. These implementations will be integrated shortly as data analysis applications of the Structural Bioinformatics Library (`http://sbl.inria.fr`). Algorithm $MST(G, D)$ has the following ingredients: **(i)** the spanning tree generator $\mathcal{R}$ returns a *maximum spanning tree*; **(ii)** the property $\Pi(\mathcal{M})$ returns true once we have computed a solution; **(iii)** $\mathcal{A}$ is the algorithm described in Section 5.2.3. Algorithm $RST(G, D)$ has the following ingredients: **(i)** $\mathcal{R}$ returns a *random spanning tree*; **(ii)** for a given parameter n_i, $\Pi(\mathcal{M})$ returns true once the algorithm has computed n_i solutions; **(iii)** $\mathcal{A}$ is also the algorithm described in Section 5.2.3. Individual calculations reported thereafter took less than one minute on a laptop computer–and therefore are not further scrutinized.

6.2 Experiments on random clusterings

We test our algorithms 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 algorithms to retrieve matchings such as the one of Figure 1, stressing the role of parameters k and D.

Random clusterings. The number of clusterings of a set Z of size t into k clusters is the number of distinct partitions of this set into k nonempty subsets. Its number is the Stirling number of second kind [13]. Adding up all these numbers yields the number of partitions of the set Z into any number of subsets, which is the Bell number $B(t)$ [10]. Such cluterings were generated using a Boltzmann sampler [8, Example 5]. Since clustering usually aims at grouping data points into a relatively small number of clusters, two pairs of parameters were used $(t = 1000, r = 20)$ and $(t = 3000, r = 50)$. 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 deriving copying F into F' and editing F', in two steps. First, we performing 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 $y = 100\tau$.

Parameters for algorithms. These 180 comparison pairs are feed to algorithms ($MST(G, D)$ and $RST(G, D)$). For both, the diameter constraint $D \in \{1, 2, 3, 4\}$.

Statistics. Since each protocol is repeated $N_r = 10$ times, we report a moustache plot of the scores Φ as well as the number of meta-clusters k, collected over the N_r repeats.

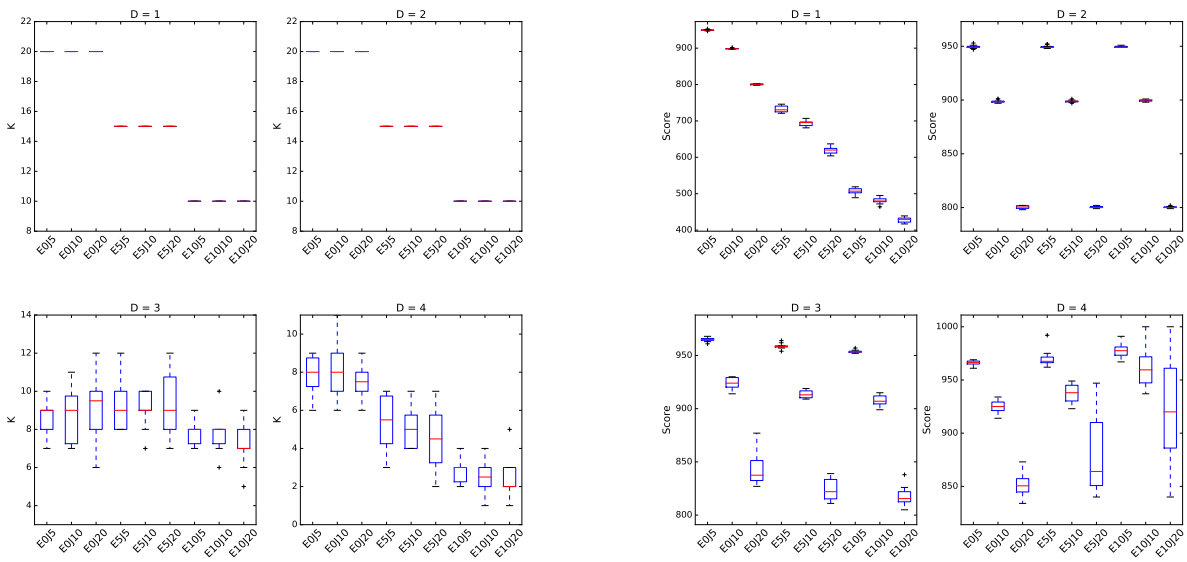
Results. Due to the lack of space, we only report the results for $(t = 1000, r = 20)$ processed by $MST(G, D)$ (Figure 3; full details in Appendix F).

For parameters $D \leq 2$ (Left panel, 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 $D = 1$ (left panel, 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 (right panel, top right) confirm that our algorithm matches the merged clusters in F' with their split counterparts in F at any jitter level. This is made clear by comparing scores for 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 (right panel, top left) decreases linearly with respect to the number of fusions bolsters this hypothesis.

For $D = 3$, our algorithm no longer discriminates between the scenarii in which we perform $r/4 = 5$ fusion operations (E5) and those in which we only jitter (E0) the original clustering. The number of meta-clusters k (left panel, bottom left) and the scores (right panel, bottom left), are near indistinguishable. There is however, an admittedly small but noticeable gap when the number of fusion operations increases to $r/2 = 10$ (E10).

For $D = 4$, the significant gaps between the number k of meta-clusters (left panel, bottom right) found across scenarii offer the possibility of guesswork regarding which ones present the most heavily edited clusterings. Although this could, as such, be an interesting application, it does not give us any relevant information on the number of clusters contained in each clustering.

Figure 3 Algorithm $MST(G, D)$ for clusterings with $(t = 1000, r = 20)$. **(Left panel)** Best value for k as a function of the 9 scenarii. **(Right panel)** Scores Φ as a function of the 9 scenarii.



6.3 Application to the instability of k-means

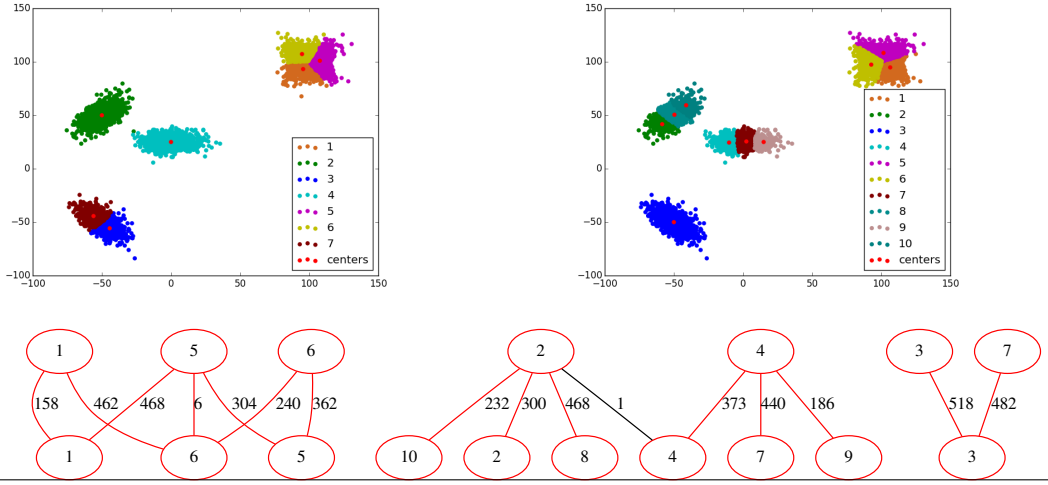
k-means is one of the most used clustering algorithms, with in particular the so-called smart seeding strategy [1], which aims at *scattering* as much as possible the centers across the dataset. However, even with this seeding strategy, k-means suffers from instabilities when the number of centers used is larger than the *exact* number of clusters. In that case, the clustering obtained depends on the initial distribution of centers withing the clusters [27].

Starting from such arbitrary clusterings, our algorithms can be used to defining meta-clusters matching one-another and infer the *correct* number of clusters for $D = 3$ (Figure 4). This case illustrates the importance of the diameter condition. Had there been no constraint on D (apart that D be finite), a similar algorithm would have found 3 meta-clusters for an optimal score of 5000, a better score at the cost of the quality of the solution.

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, with however polynomial time dynamic programming algorithms for specific graphs (in particular trees). These algorithms can in turn be used to design efficient (exact, approximation, and heuristic) algorithms, based on spanning trees, for general graphs. In the spirit of Lemma 8 (proving the existence of at least one spanning tree T of G such that an optimal solution for the family-matching problem for G constrained by T gives an optimal solution for the family-matching problem for G) we 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 an approximation for the family-matching problem for G.

Figure 4 Detecting the instability of k-means with smart seeding: illustration on a 2D point set with 5k points, drawn according to a mixture of 5 gaussians. (Top left) k-means with $k = 7$. **(Top right)** k-means with $k = 10$. When the number of centers passed is larger than the exact number of clusters, four here, these clusters get split arbitrarily. **(Bottom panel)** The result of $RST(G, D)$ run with $D = 3$, and stopped after $n_i = 10000$ iterations superimposed on the intersection graph of the two clusterings. The cluster labels follow the legends on the top panels. The meta-clusters are reported in red. Using these two clusterings, our algorithm recovers the 4 clusters. The solution score is $\Phi = 4999$, which stems from the fact one point is assigned to the wrong cluster in the top left clustering (cyan point in green cluster).



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-cluster 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] E. Bannai and T. Ito. On finite Moore graphs. *J. Fac. Sci. Univ. Tokyo Sect. IA Math.*, 20:191–208, 1973.

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

[8] 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.

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

[10] P. Flajolet and R. Sedgewick. *Analytic combinatorics.* cambridge University press, 2009.

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

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

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

[14] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182(1):105–142, 1999.

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

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

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

[18] J. Kleinberg. An impossibility theorem for clustering. *Advances in neural information processing systems*, pages 463–470, 2003.

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

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

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

[25] 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.

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

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

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

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

[30] 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 - Table of notations

Notation	Definition
$Z = \{z_1, \dots, z_t\}$	Set of $t \geq 1$ elements
$F = \{F_1, \dots, F_r\}$	Family of $r \geq 1$ disjoint subsets of Z
$F' = \{F'_1, \dots, F'_{r'}\}$	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
$\mathcal{S} = \{S_1, \dots, S_k\}$	(k, D)-family-matching
$\Phi(\mathcal{S}) = \sum_{i=1}^{k} \sum_{e \in E(G[S_i])} w_e$	Score of a (k, D)-family-matching $\mathcal{S}$
$\mathcal{S}(G, k, D)$	Set of all (k, D)-family-matching
$\Phi_{k,D}(G) = \max_{\mathcal{S} \in \mathcal{S}(G,k,D)} \Phi(\mathcal{S})$	Optimal score for the (k, D)-family-matching problem
$\Phi_D(G) = \max_{k \in \{0,\dots,n\}} \Phi_{k,D}(G)$	Optimal score for the D-family-matching problem
$\mathcal{S}_{k,D}(G, T_r)$	Set of all (k, D)-family-matching constrained by T_r
$\Phi_{k,D}(G, T_r) = \max_{\mathcal{S} \in \mathcal{S}_{k,D}(G,T_r)} \Phi(\mathcal{S})$	Optimal score for the (k, D)-family-matching problem constrained by T_r
$\Phi_D(G, T_r) = \max_{k \in \{0,\dots,n\}} \Phi_{k,D}(G, T_r)$	Optimal score for the D-family-matching problem constrained by T_r

B Appendix - Equivalent definitions of the family-matching problems

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 $Z = \{z_1, \dots, z_t\}$ with $t = w_{u_1,u'_1}$. Set $F = \{F_1\}$ with $F_1 = \{z_1, \dots, z_t\}$ and set $F' = \{F'_1\}$ with $F'_1 = \{z_1, \dots, z_t\}$. 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 $Z^{G'} = \{z_1, \dots, z_t\}$, $F^{G'} = \{F_1, \dots, F_r\}$, and $F'^{G'} = \{F'_1, \dots, F'_{r'}\}$. Without loss of generality, assume that $x \in U$. Let $N_G(x) = \{u'_1, \dots, u'_{d_x}\}$, 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 $i \in \{1, \dots, d_x\}$ (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 $Z = Z^{G'} \cup \{z_{t+1}, \dots, z_{t+w_x}\} = \{z_1, \dots, z_t, z_{t+1}, \dots, z_{t+w_x}\}$. Set $F = \{F_1, \dots, F_r, F_{r+1}\}$, where $F_{r+1} = \{z_{t+1}, \dots, z_{t+w_x}\}$. For every $i, j \in \{1, \dots, d_x\}$, $i \neq j$, let $X_i \subseteq \{z_{t+1}, \dots, z_{t+w_x}\}$ with $|X_i| = w_{x,u'_i}$ and such that $X_i \cap X_j = \emptyset$. Finally, set $F' = \{F''_1, \dots, F''_{r'}\}$, where $F''_i = F'_i \cup X_i$ for every $i \in \{1, \dots, d_x\}$, and $F''_i = F'_i$ for every $i \in \{d_x + 1, \dots, r'\}$. 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 (k, D)-family-matching.

Definition. 7 ((k,D)-family-matching)**.** *Let $k, D \geq 1$ be two positive integers. A (k, D)-family-matching is a family $\mathcal{P} = \{P_1, \ldots, P_k\}$ 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 (k, 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 $\mathcal{P}_{k,D}(F, F')$ be the set of all (k, D)-family-matching for F, F', k, and D. We now formalize an equivalent definition of the (k, D)-family-matching problem.

Definition. 8 ((k,D)-family-matching problem)**.** *Let $k, D \geq 1$ be two positive integers. The (k, D)-family-matching problem consists in determining a (k, D)-family-matching that maximizes the score f. Formally, we aim at computing:*

$$f_{k,D}(F, F') = \max_{\mathcal{P} \in \mathcal{P}_{k,D}(F,F')} f(\mathcal{P}).$$

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

Property 3. *Let $k, D \geq 1$ be any two integers. Let $L \geq 0$ be any positive real number. Consider any instance of the (k, D)-family-matching problem defined by Z, F, and F', and consider the associated intersection graph G. Then, there is a (k, D)-family-matching $\mathcal{P}$ for Z, F, and F', such that $f(\mathcal{P}) \geq L$ if and only if there is a (k, D)-family-matching $\mathcal{S}$ of G such that $\Phi(\mathcal{S}) \geq L$.*

We define similarly the equivalent definition of the D-family-matching problem.

C Appendix - The family-matching problems are very hard to solve

C.1 NP-completeness results

We prove that the (k, D)-family-matching problem, $D \geq 2$, and the 2-family-matching problem are NP-complete. In our reduction, we use set packing problem, a well known NP-complete problem [17]. 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 $i \in \{1, \ldots, p\}$.

C.1.1 NP-completeness of the (k,D)-family-matching problem

This section is devoted to prove Theorem 1. 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 every $i \in \{1, \ldots, p\}$. We first construct the intersection graph G of the (k, D)-family-matching problem (Definition 9).

Definition. 9 (Construction of the intersection graph G for the (k,D)-family-matching problem)**.** *The intersection graph $G = (U, U', E, w)$ is defined as follows. Let $\delta_u = \lfloor (D-1)/2 \rfloor$ and $\delta_{u'} = \lceil (D-1)/2 \rceil$. Let B be any real number such that $B > p(\lceil D/2 \rceil + 3)$.*

- *Set $U = U_0 \cup U_1 \cup \ldots \cup U_{\delta_u}$, where*
  - *$U_j = \{u_1^j, \ldots, u_p^j\}$ for every $j \in \{1, \ldots, \delta_u\}$*
  - *and the set $U_0 = \{u_1, \ldots, u_p\}$ corresponds to Y.*
- *Set $U' = U'_0 \cup U'_1 \cup \ldots \cup U'_{\delta_{u'}}$, where*
  - *$U'_j = \{u_1'^j, \ldots, u_p'^j\}$ for every $j \in \{1, \ldots, \delta_{u'}\}$*
  - *and the set $U'_0 = \{u'_1, \ldots, u'_t\}$ corresponds to X.*
- *Set $E = E' \cup E'' \cup E^{1,1} \cup \ldots \cup E^{\delta_u, \delta_u} \cup E^{1,2} \cup \ldots \cup E^{\delta_{u'}-1, \delta_{u'}}$, where*
  - *$E' = \{\{u_i, u'_j\} \mid x_j \in Y_i, 1 \leq i \leq p, 1 \leq j \leq t\}$,*
  - *$E'' = \{\{u_1, u_1'^1\}, \ldots, \{u_p, u_p'^1\}\}$,*
  - *$E^{i,i} = \{\{u_1^i, u_1'^i\}, \ldots, \{u_p^i, u_p'^i\}\}$ for every $i \in \{1, \ldots, \delta_u\}$,*
  - *and $E^{i-1,i} = \{\{u_1^{i-1}, u_1'^i\}, \ldots, \{u_p^{i-1}, u_p'^i\}\}$ for every $i \in \{2, \ldots, \delta_{u'}\}$.*
- *Set $w_e = 1$ for every $e \in E \setminus E^{\delta_u, \delta_{u'}}$ and $w_e = B$ for every $e \in E^{\delta_u, \delta_{u'}}$.*

In the rest of the section, we assume that $w_{v,v'} = 0$ for every two nodes v, v' such that $\{v, v'\} \notin E$.

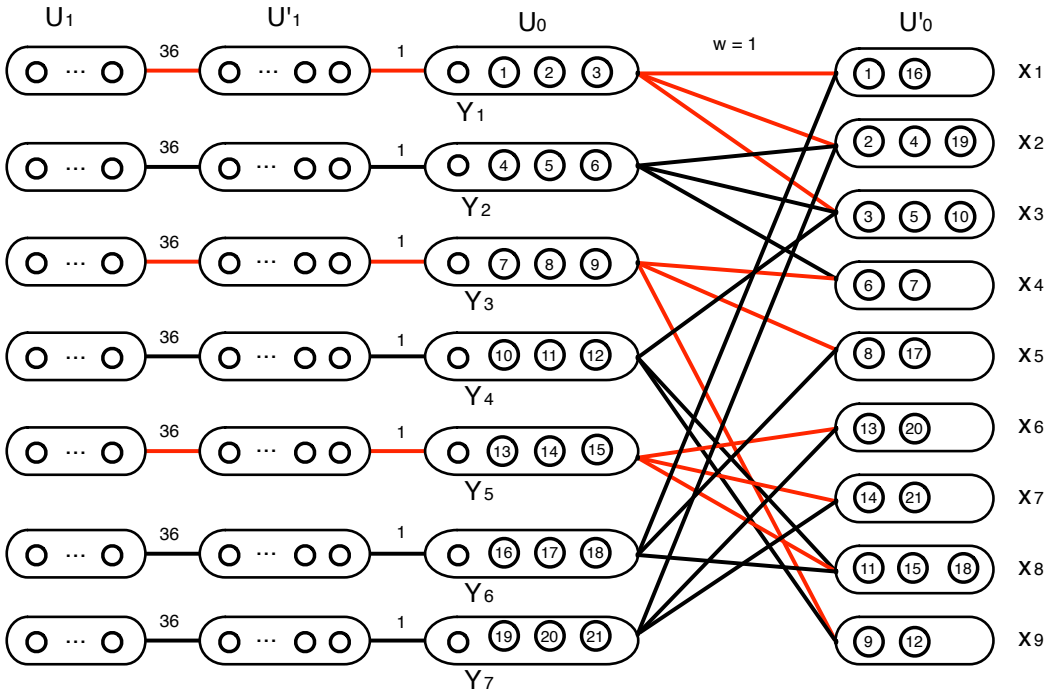
Lemma. 9. *If there is a solution $\mathcal{C}$ for the instance $\mathcal{I}_{sp}$ of set packing problem such that $|\mathcal{C}| = k$, then there is a solution $\mathcal{S}$ for the (k, D)-family-matching problem for G such that $\Phi(\mathcal{S}) = k(B + D + 1)$.*

Proof of Lemma 9. 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 (k, D)-family-matching problem for G such that $\Phi(\mathcal{S}) = k(B+D+1)$. Assume that $\mathcal{C} = \{Y_1, \ldots, Y_k\}$ (we permute the indices otherwise). Let $\mathcal{S} = \{S_1, \ldots, S_k\}$ be such that $S_i = N_G(u_i) \cup \{u_i, u_i^1, \ldots, u_i^{\delta_u}, u_i'^1, \ldots, u_i'^{\delta_{u'}}\}$ for every $i \in \{1, \ldots, k\}$. The sets are disjoint. In other words, for every $i, j \in \{1, \ldots, k\}$, $i \neq j$, then $S_i \cap S_j = \emptyset$ because $\mathcal{C}$ is a set packing and, by construction of G, we have $N_G(u_i) \cap N_G(u_j) = \emptyset$. Furthermore, for every $i \in \{1, \ldots, k\}$, the diameter of $G[S_i]$ is D because the graph $G[(S_i \setminus N_G(u_i)) \cup \{u_i^1\}]$ is a path, denoted H_i, composed of $D-1$ nodes and because the three nodes of $N_G(u_i)) \setminus \{u_i^1\}$ are neighbors of u_i. Finally, we get

$$\Phi(\mathcal{S}) = 4k + \sum_{i=1}^{k} \sum_{a \in V(H_i)} \sum_{b \in V(H_i)} w_{a,b} = 3k + (D-2)k + Bk = k(B + D + 1).$$

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

Figure 5 Illustration of the proof of Theorem 1 with D = 3 and k = 3. See details in the text.



Lemma. 10. *If there is a solution $\mathcal{S}$ for the (k, D)-family-matching problem for G such that $\Phi(\mathcal{S}) = k(B + D + 1)$, then there is a solution $\mathcal{C}$ for the instance $\mathcal{I}_{sp}$ of set packing problem such that $|\mathcal{C}| = k$.*

Proof of Lemma 10. Consider any solution $\mathcal{S} = \{S_1, \ldots, S_k\}$ for the (k, D)-family-matching problem for G such that $\Phi(\mathcal{S}) = k(B + D + 1)$. We prove that there exists a solution $\mathcal{C}$ for the instance $\mathcal{I}_{sp}$ of set packing problem such that $|\mathcal{C}| = k$. We first prove that, for every $i \in \{1, \ldots, k\}$, we have

$$\sum_{a \in S_i} \sum_{b \in S_i} w_{\{a,b\}} = B + D + 1.$$

To do that, we show that any sub-graph H of G with diameter at most D is such that

$$\sum_{e \in E(H)} w_e \leq B + D + 1.$$

Observe that the graph $G[(U \cup V) \setminus V']$ is composed of p disjoint paths each composed of D nodes. Let $\{H_1, \ldots, H_p\}$ be the p disjoint paths of $G[(U \cup V) \setminus V']$. Consider any sub-graph H of G with diameter at most D. Let h_i be the number of nodes of H_i that are in H for every $i \in \{1, \ldots, p\}$.

First, suppose that $h_i \leq D - 1$ for every $i \in \{1, \ldots, p\}$. Since H has diameter at most D, then we necessarily have $h_i + h_j \leq D$ for every $i, j \in \{1, \ldots, p\}$, $i \neq j$. Indeed, the distance between any node of H_i and any node of H_j is at least 2, for every $i, j \in \{1, \ldots, p\}$, $i \neq j$. Thus,

if $h_i + h_j > D$, then the diameter of H is at least $D + 1$. We get that

$$\sum_{i=1}^{p} h_i \leq p\lceil D/2 \rceil.$$

Even if we assume that all nodes of V' are in H, we get that

$$\sum_{e \in E(H)} w_e \leq p\lceil D/2 \rceil + 3p < B.$$

Second, assume that there exists $i \in \{1, \ldots, p\}$ such that $h_i = D$. Then, it necessarily means that $V(H) \subseteq V(H_i) \cup N_G[u_i]$ because H_i is a path composed of D nodes. We proved the result because, in the worst case, the sum of the weights is $B + D + 1$.

By previous claims, we know that for every $i \in \{1, \ldots, k\}$, we have

$$\sum_{a \in S_i} \sum_{b \in S_i} w_{\{a,b\}} = B + D + 1$$

because $\Phi(\mathcal{S}) = k(B + D + 1)$. Without loss of generality, assume that $u_i \in S_i$ for every $i \in \{1, \ldots, k\}$ (we permute the indices otherwise). By previous remarks, we have $S_i = N_G(u_i) \cup \{u_i, u_i^1, \ldots, u_i^{\delta_u}, u_i'^1, \ldots, v_i^{\delta_{u'}}\}$ for every $i \in \{1, \ldots, k\}$ because $\Phi(\mathcal{S}) = k(B + D + 1)$. We have $S_i \cap S_j = \emptyset$ for every $i, j \in \{1, \ldots, k\}$, $i \neq j$, and so $N_G(u_i) \cap N_G(u_j) = \emptyset$. By construction of G, we get that $Y_i \cap Y_j = \emptyset$ and so $\mathcal{C} = \{Y_1, \ldots, Y_k\}$ is a set packing for the instance $\mathcal{I}_{sp}$ such that $|\mathcal{C}| = k$. □

We are now able to prove Theorem 1.

Proof of Theorem 1. First, the reduction (Definition 9) can be clearly done in polynomial time. Second, the decision version of the (k, D)-family-matching is in NP because, given any $\mathcal{S}$, one can check in polynomial time if $\mathcal{S}$ is a (k, D)-family-matching and one can compute in polynomial time the score $\Phi(\mathcal{S})$. Finally, Lemmas 9 and 10 prove that there is a solution $\mathcal{C}$ for the instance $\mathcal{I}_{sp}$ of set packing problem such that $|\mathcal{C}| = k$ if and only if there is a solution $\mathcal{S}$ for the (k, D)-family-matching problem for G such that $\Phi(\mathcal{S}) = k(B + D + 1)$. Thus, the decision version of the (k, D)-family-matching is NP-complete. □

To illustrate the proof of Theorem 1, 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 $Y_1 = \{x_1, x_2, x_3\}$, $Y_2 = \{x_2, x_3, x_4\}$, $Y_3 = \{x_4, x_5, x_9\}$, $Y_4 = \{x_3, x_8, x_9\}$, $Y_5 = \{x_6, x_7, x_8\}$, $Y_6 = \{x_1, x_5, x_8\}$, $Y_7 = \{x_2, x_6, x_7\}$. Let $k = 3$ and $D = 3$. We set $B = p(\lceil D/2 \rceil + 3) + 1 = 36$. The graph G depicted in Figure 5 is the graph obtained from Definition 9. A possible valid instance of the (k, D)-family-matching problem in terms of Z, F, and F', is also depicted in Figure 5. There is a set packing $\mathcal{C} = \{Y_1, Y_3, Y_5\}$ of size 3 and there is a (k, D)-family-matching $\mathcal{S}$ such that $\Phi(\mathcal{S}) = 120$ (depicted in red in Figure 5).

C.1.2 NP-completeness of the 2-family-matching problem

This section is devoted to prove Theorem 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 $i \in \{1, \ldots, p\}$. We first construct the intersection graph G of the 2-family-matching problem (Definition 10).

Definition. 10 (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*
  - $U_1 = \{u_1^1, \dots, u_p^1\}$ *corresponds to* Y
  - *and* $U_2 = \{u_1^2, \dots, u_p^2\}$.
- *Set $U' = U'_1 \cup U'_2$, where*
  - $U'_1 = \{u_1'^1, \dots, u_t'^1\}$ *corresponds to* X
  - *and* $U'_2 = \{u_1'^2, \dots, u_p'^2\}$.
- *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 $e \in E_c$.*

Lemma. 11. *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 11. 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 $\mathcal{C} = \{Y_1, \dots, Y_k\}$ (we permute the indices otherwise). Let $\mathcal{S} = \{S_1, \dots, S_p\}$, where $S_i = \{u_i^1\} \cup N_G(u_i^1)$ for every $i \in \{1, \dots, k\}$ and $S_i = \{u_i^1, u_i'^2, u_i^2\}$ for every $i \in \{k+1, \dots, p\}$. The sets are disjoint. In other words, for every $i, j \in \{1, \dots, p\}$, $i \neq j$, 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 $i', j' \in \{1, \dots, k\}$, $i' \neq j'$. Furthermore, for every $i \in \{1, \dots, p\}$, the diameter of $G[S_i]$ is at most 2. Finally, we get

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

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. 12. *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 12. 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 4. *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 4. 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 4, 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).

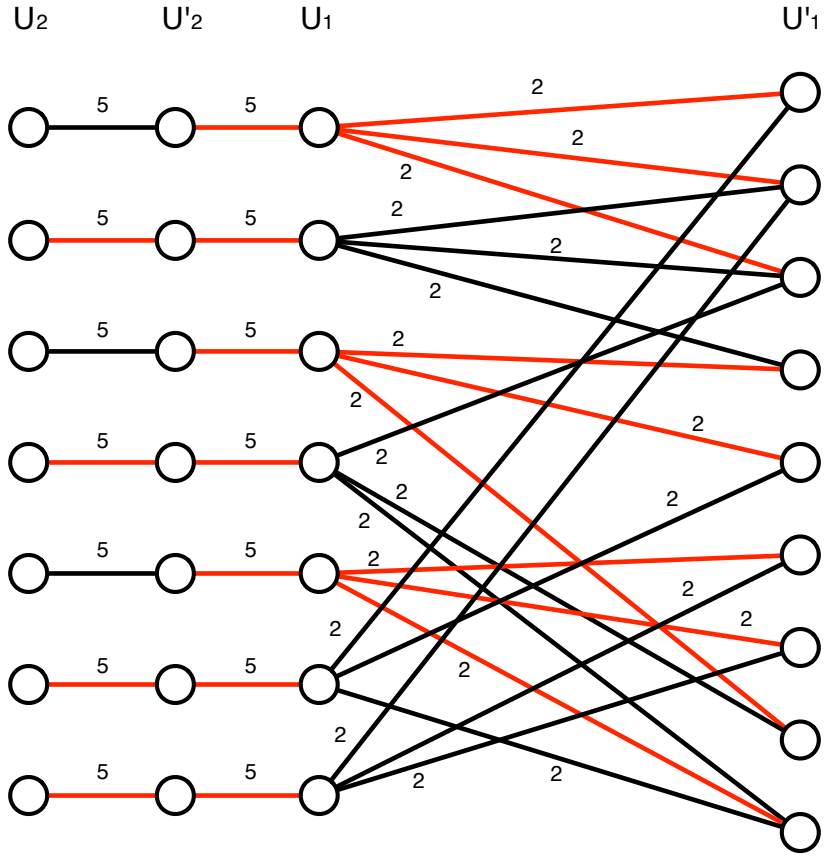
We deduce the following claim.

Claim 5. *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 5, 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 $i, i' \in \{i_1, \ldots, i_k\}$, $i \neq i'$. 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 Theorem 2.

Figure 6 Illustration of the proof of Theorem 2. See details in the text.



Proof of Theorem 2. First, the reduction (Definition 10) 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 11 and 12 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 illustrate the proof of Theorem 2, 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 6 is the graph obtained from Definition 10. 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 6).

C.2 Hardness of approximation for general graph even with k = D = 1

Proof of Lemma 3. Since $k = 1$, then $\mathcal{S} = \{S_1\}$. Since $D = 1$, then it necessarily means that the graph $G[S_1]$ induced by the set of nodes S_1 is a complete graph (for every $v, v' \in V$, $v \neq v'$, then $\{v, v'\} \in E(G[S_1])$). Recall that $w_e = 1$ for every $e \in E$. The problem of computing such a set $\mathcal{S}$ maximizing $\Phi(\mathcal{S})$ ((1, 1)-family-matching problem) is equivalent to Clique problem. Indeed, for any real $g > 0$, there exists a set $\mathcal{S}$ such that $\Phi(\mathcal{S}) \geq g(g-1)/2$ if and only if there exists a complete graph composed of at least g nodes in G. The hardness of approximation of $(1, 1)$-family-matching problem is directly deduced from the hardness of approximation of Clique problem [14]. □

C.3 Unbounded ratio between scores by increasing the diameter by one

Proof of Lemma 4. Let $t = n - 1$. Let $Z = \{z_1, \ldots, z_t\}$, $F = \{F_1\}$, and $F' = \{F'_1, \ldots, F'_t\}$ be an instance of the (k, D)-family-matching problem, where $F_1 = \{z_1, \ldots, z_t\}$ and $F'_i = \{z_i\}$ for every $i \in \{1, \ldots, t\}$. The intersection graph $G = (U, U', E, w)$ is such that $U = \{u_1\}$, $U' = \{u'_1, \ldots, u'_t\}$, $E = \{\{u_1, u'_i\} \mid 1 \leq i \leq t\}$, and $w_e = 1$ for every $e \in E$. We first prove that any solution $\mathcal{S}^{D=1}$ for the $(k, 1)$-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 $\Phi(\mathcal{S}^{D=2}) = t = n - 1$. □

C.4 Optimizing first the score of a single set can be arbitrarily bad

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

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

Observe that the graph G is bipartite.

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

We finally prove that there exists a $(\lambda, 2)$-family-matching $\mathcal{S}$ for G such that $\Phi(\mathcal{S}) \geq \lambda(\lambda - 2)$. Let $\mathcal{S} = \{S_1, \ldots, S_\lambda\}$ be such that $S_i = \{u_i\} \cup U'^i$ for every $i \in \{1, \ldots, \lambda\}$. Observe that

$S_i \cap S_j = \emptyset$ for every $i, j \in \{1, \ldots, \lambda\}$, $i \neq j$. Furthermore, the graph $G[S_i]$ is a star and so has diameter 2. Thus, $\mathcal{S}$ is a $(\lambda, 2)$-family-matching for G. The number of edges of $G[S_i]$ is $|E(G[S_i])| = \lambda - 2$ for every $i \in \{1, \ldots, \lambda\}$. Since $w_e = 1$ for every $e \in E$, we finally get that

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

This concludes the proof of Lemma 5. □

D Appendix - Polynomial time dynamic programming algorithms for some classes

D.1 The (k,D)-family-matching problem for trees

We prove in Lemma 13 an exact dynamic programming algorithm for trees.

Lemma. 13 (Computation of $\Phi_{k,D}(G)$ for trees)**.** *Let $k, D \geq 1$ be two positive integers. Consider any intersection tree $G = T = (V, E, w)$ of maximum degree $\Delta \geq 0$. Then, there exists an $O(2^{\Delta} k^{\Delta} D^2 \Delta^2 n)$-time complexity algorithm for the (k, D)-family-matching problem for G.*

Proof of Lemma 13. 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 $\Psi_{k',D}$ as follows. For every $v \in V$, every $k' \in \{0, \ldots, k\}$, and every $i \in \{-1, 0, \ldots, D\}$, then $\Psi_{k',D}(T_v, i)$ is the score of an optimal solution $\mathcal{S}$ for the (k', 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 $\Psi_{k',D}(T_v, 0)$ is the score of an optimal solution $\mathcal{S}$ when $\{v\} \in \mathcal{S}$ (say otherwise, v is alone a set). In the following, we abuse the notation writing $\Psi_{k',D,(}(v, i)$ instead of $\Psi_{k',D}(T_v, i)$. By convention, if there is no admissible solution, we set $\Psi_{k',D,(}(v, i) = -\infty$. This is the case when, for instance, $|V(T_v)| < k'$.

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

- $\Psi_{0,D}(v, i) = \Psi_{1,D}(v, i) = 0$ for every $i \in \{-1, 0, \ldots, D\}$,
- $\Psi_{k',D}(v, i) = -\infty$ for every $k' \in \{2, \ldots, k\}$ and every $i \in \{-1, 0, \ldots, D\}$.

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 $\Psi_{k',D}(v_j, i)$ for every $j \in \{1, \ldots, q\}$, every $k' \in \{0, \ldots, k\}$, and every $i \in \{-1, \ldots, D\}$. We prove that we can compute $\Psi_{k',D}(v, i)$ for every $k' \in \{0, \ldots, k\}$ and every $i \in \{-1, \ldots, D\}$ in $O(2^{\Delta} k^{\Delta} D^2 \Delta^2)$-time. There are two different cases (corresponding to the two following claims).

Claim 6. *For every $k' \in \{0, \ldots, k\}$ and every $i \in \{-1, 0\}$, then*

$$\Psi_{k',D}(v, i) = \max_{(k_1, \ldots, k_q) \in \mathcal{K}} \left(\sum_{j=1}^{q} \max_{i' \in \{-1, \ldots, D\}} \Psi_{k_j,D}(v_j, i') \right),$$

where $\mathcal{K}$ is the set of all vectors $(k_1, \ldots, k_q)$ such that

- $0 \leq k_j \leq k'$ *for every* $j \in \{1, \ldots, q\}$,
- $k_j \leq |T_{v_j}|$ *for every* $j \in \{1, \ldots, q\}$,
- $\sum_{j=1}^{q} k_j = k' - i - 1$.

Proof of Claim 6. Let us first consider $i = -1$. We consider here an optimal solution $\mathcal{S}$ for the (k', D)-family-matching problem for T_v such that $\{v\} \notin S$ for every $S \in \mathcal{S}$. For every $j \in \{1, \ldots, q\}$ and for every possible number of sets k_j that belong to T_{v_j}, we compute an optimal solution for the (k_j, D)-family-matching problem (whatever the value of i). Say otherwise, we compute $\max_{i' \in \{-1, \ldots, D\}} \Psi_{k_j, D}(v_j, i')$. We then choose $(k_1, \ldots, k_q) \in K$ such that $\sum_{j=1}^{q} \max_{i' \in \{-1, \ldots, D\}} \Psi_{k_j, D}(v_j, i')$ is maximum.

Let us now consider $i = 0$. It means that we consider an optimal solution $\mathcal{S}$ for the (k', D)-family-matching problem for T_v such that $\{v\} \in \mathcal{S}$. We do exactly the same reasoning than before. Note that $\sum_{j=1}^{q} k_j = k' - 1$ because $i = 0$. □

Claim 7. *For every* $k' \in \{0, \ldots, k\}$, *and every* $i \in \{1, \ldots, D\}$, *then*

$$\Psi_{k', D}(v, i) = \max_{(b_1, \ldots, b_q) \in \{0,1\}^q} \left(\max_{(k_1, \ldots, k_q) \in \mathcal{K}} \left(\max_{j \in \{1, \ldots, q\}, b_j = 1} X_1 + X_2 + X_3 \right) \right),$$

where

$$X_1 = \Psi_{k_j, D}(v_j, i-1) + w_{v, v_j},$$

$$X_2 = \sum_{j' \in \{1, \ldots, q\} \setminus \{j\}, b_{j'} = 1} \max_{i' \in \{0, \ldots, D-i-1\}} \Psi_{k_j, D}(v_{j'}, i') + w(v, v_{j'}),$$

$$X_3 = \sum_{j' \in \{1, \ldots, q\} \setminus \{j\}, b_{j'} = 0} \max_{i' \in \{-1, \ldots, D\}} \Psi_{k_j, D}(v_{j'}, i'),$$

and where $\mathcal{K}$ *is the set of all vectors* $(k_1, \ldots, k_q)$ *such that*

- $b_j \leq k_j \leq k'$ *for every* $j \in \{1, \ldots, q\}$,
- $k_j \leq |T_{v_j}|$ *for every* $j \in \{1, \ldots, q\}$,
- $\sum_{j=1}^{q} k_j = k' + \sum_{j=1}^{q} b_j - 1$.

Proof of Claim 7. The value $\Psi_{k', D}(v, i)$ corresponds to the score of an optimal solution $\mathcal{S}$ for the (k', D)-family-matching problem for T_v such that, if $i \geq 0$, then there exists $S_v \in \mathcal{S}$, $v \in S_v$, and the sub-tree induced by the of nodes S has depth at most i. To compute $\Psi_{k', D}(v, i)$, we enumerate all the possible solutions as follows. For every sub-tree T_{v_j}, $1 \leq j \leq q$, either $S_v \cap V(T_{v_j}) \neq \emptyset$ or $S_v \cap V(T_{v_j}) = \emptyset$. This corresponds to $b_j = 1$ or $b_j = 0$, respectively. Then, we enumerate all the possible number of sets k_j that are in T_{v_j}, for every $j \in \{1, \ldots, q\}$. Finally there is at least one sub-tree T_{v_j} such that $S_v \cap V(T_{v_j}) \neq \emptyset$ and such that the sub-tree induced by the nodes $S_v \cap V(T_{v_j})$ has depth at most $i - 1$. We enumerate the q possible choices for such a sub-tree. Now, for every b_j, k_j, and choice for the sub-tree of depth at most $i - 1$ ($1 \leq j \leq q$), that satisfy the previous properties, we compute an optimal solution. We prove that $X_1 + X_2 + X_3$ is the score of such an optimal solution for any possible choice.

- Computation of X_1. Let T_{v_j}, for some $j \in \{1, \ldots, q\}$, be a sub-tree such that $S_v \cap V(T_{v_j}) \neq \emptyset$ and such that the sub-tree induced by the nodes $S_v \cap V(T_{v_j})$ has depth at most $i - 1$. Since k_j is fixed, we consider the score of an optimal solution for the (k_j, D)-family-matching problem for T_{v_j} such that v_j is in a sub-tree (set) of depth at most $i - 1$. Thus, X_1 is the sum of such a score and the weight w_{v, v_j} of edge $\{v, v_j\}$ because T_j has been chosen in order to form a sub-tree (in the solution for T_v) that contains v and with depth at most i.

- Computation of X_2. We consider all the sub-trees $T_{v_{j'}} \neq T_{v_j}$ such that $S_v \cap V(T_{v_{j'}}) \neq \emptyset$. Thus, for every $j' \in \{1, \ldots, q\}$, $j' \neq j$, such that $b_{j'} = 1$, we compute an optimal solution for the $(k_{j'}, D)$-family-matching problem for $T_{v_{j'}}$ such that the sub-tree containing v' has depth at most $D - i - 1$. Note that if the depth $i' > D - i - 1$, then the sub-tree induced by S_v has diameter at least $D + 1$. Thus, X_2 is the sum of the scores of such optimal solutions plus the sum of the weights of the edges between the different roots and v, that is $\sum_{j' \in \{1,\ldots,q\} \setminus \{j\}, b_{j'}=1} w(v, v_{j'})$.
- Computation of X_3. We consider all the sub-trees $T_{v_{j'}}$ such that $S_v \cap V(T_{v_{j'}}) = \emptyset$. For every $j' \in \{1, \ldots, q\}$, such that $b_{j'} = 0$, we compute an optimal solution for the $(k_{j'}, D)$-family-matching problem for $T_{v_{j'}}$. Observe that the depth has no importance here because $\{v, v_{j'}\}$ is not in the solutions considered here. Thus, X_3 is the sum of the scores of such optimal solutions.

Finally, $\Psi_{k',D}(v, i)$ is the largest score among all optimal solutions for all possible choices of the previous parameters. □

For every $v \in V$, we address the time complexity of computing Ψ (for all possible values of k and i) as follows. The time complexity of the computation done in Claim 6 is $O(k^{q+1}qD)$. The time complexity of the computation done in Claim 7 is $O(2^q k^{q+1} q^2 D^2)$. Since T is rooted at node r and r has degree at most $\Delta - 1$, we have $q \leq \Delta - 1$ for every $v \in V$. We get that the time complexity of the dynamic programming algorithm is $O(2^\Delta k^\Delta D^2 \Delta^2 n)$.

To conclude the proof of Lemma 13, when we have computed $\Psi_{k',D}(r, i)$ for every $k' \in \{0, \ldots, k\}$ and every $i \in \{-1, \ldots, D\}$, we can deduce an optimal solution $\mathcal{S}$ and the optimal value of the (k, D)-family-matching problem for T. Indeed,

$$\Phi(\mathcal{S}) = \max_{i \in \{-1,\ldots,D\}} \Psi_{k,D}(r, i).$$

Recall that $i = -1$ means that node r does not belong to any set of the solution. □

We deduce in Corollary 2 a polynomial time algorithm for trees of bounded maximum degrees.

Corollary. 2. *Let $k, D \geq 1$ be two positive integers. Consider any intersection tree $G = T = (V, E, w)$ of bounded maximum degree $\Delta = O(1)$. Then, there exists a polynomial time algorithm for the (k, D)-family-matching problem for G.*

D.2 The D-family-matching problem for trees

Claim 8. *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 8. 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 (k', 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 9. *For every $i \in \{1, \dots, D\}$, then*

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

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

Proof of Claim 9. 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 $j \in \{1, \dots, q\}$, such that the set (sub-tree) that contains v_j in the solution for T_v, is such that the sub-tree induced by $S_v \cap V(T_{v_j})$ has depth $i-1$. In order to compute such j, we enumerate the q different possibilities. For every possible choice ($j = 1, \dots, q$), we compute the largest possible score. Such a score is $\Psi_D(v_j, i-1)$ plus the the weight w_{v,v_j} of the edge $\{v, v_j\}$ plus the largest possible score for the other neighbors of v. More precisely, for every $j' \in \{1, \dots, q\}$, $j' \neq j$, there are two cases.

- $S_v \cap V(T_{v_j}) = \emptyset$. In that case, the largest possible score corresponding to the sub-tree $T_{v_{j'}}$ is $\max_{i' \in \{1,\dots,D\}} \Psi_D(v_{j'}, i')$.
- $S_v \cap V(T_{v_j}) \neq \emptyset$. In that case, the largest possible score is $\max_{i' \in \{1,\dots,D-i-1\}} \Psi_D(v_{j'}, i') + w_{v,v_{j'}}$. 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 Lemma 6 is polynomial because $\Delta \leq n-1$ and $D \leq n-1$.

D.3 The (k,D)-family-matching problem for paths

In this section, we illustrate the result of Lemma 13 by considering paths. Consider an intersection path $G = (V, E, w)$. By Lemma 13, given $k, D \geq 1$, there is an $O(k^2 D^2 n)$-time complexity algorithm for the (k, D)-family-matching problem because $\Delta = 2$. We prove in Lemma 14 a better time complexity algorithm for the (k, D)-family-matching problem. Indeed, the time complexity is $O(kDn)$.

Lemma. 14 (Computation of $\Phi_{k,D}(G)$ for paths)**.** *Let $k, D \geq 1$ be two positive integers. Consider any intersection path $G = (V, E, w)$. Then, there exists an $O(kDn)$-time complexity algorithm for the (k, D)-family-matching problem for G.*

Proof of Lemma 14. Let $V = \{v_1, \dots, v_n\}$. Let $E = \{\{v_j, v_{j+1}\} \mid 1 \leq j \leq n-1\}$. We define the function $\Psi_{k',D}$ as follows. For any $t \in \{1, \dots, n\}$, any $k' \in \{0, \dots, k\}$, and any $i \in \{\max(1, t-D), \dots, t+1\}$, then $\Psi_{k',D}(v_t, i)$ is the score of an optimal solution $\mathcal{S}$ of the (k', D)-family-matching problem, for the sub-path induced by the set of nodes $\{v_1, \dots, v_t\}$, such that $\{v_i, \dots, v_t\} \in \mathcal{S}$. The case $i = t+1$ means that v_t does not belong to any set. Note that we consider $i \geq \max(1, t-D)$ because, otherwise we would not have an admissible solution (because of the diameter constraint). By convention, if there is no admissible solution, we set $\Psi_{k',D}(v_t, i) = -\infty$. This is the case when, for instance, $k' > t$.

First of all,

- $\Psi_{1,D}(v_1, 1) = \Psi_{0,D}(v_1, 2) = 0$,
- $\Psi_{0,D}(v_1, 1) = \Psi_{1,D}(v_1, 2) = -\infty$,
- $\Psi_{k',D}(v_1, i) = -\infty$ for every $k' \in \{2, \ldots, k\}$ and every $i \in \{1, 2\}$.

Let $t \in \{1, \ldots, n-1\}$. Suppose we have computed $\Psi_{k',D}(v_{t'}, i)$ for every $t' \in \{1, \ldots, t\}$, every $k' \in \{0, \ldots, k\}$, and every $i \in \{\max(1, t-D), \ldots, t'+1\}$. We prove that we can compute $\Psi_{k',D}(v_{t'+1}, i)$ for every $k' \in \{0, \ldots, k\}$ and every $i \in \{\max(1, t'+1-D), \ldots, t'+1\}$ in $O(Dk)$-time. There are three different cases (corresponding to the three following claims).

Claim 10. *For every $k' \in \{0, \ldots, k\}$ and every $i \in \{\max(1, t+1-D), \ldots, t\}$, then*

$$\Psi_{k',D}(v_{t+1}, i) = w_{v_t, v_{t+1}} + \Psi_{k',D}(v_t, i).$$

Proof of Claim 10. 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 an optimal solution for the (k', D)-family-matching problem, for the sub-path induced by the set of nodes $\{v_1, \ldots, v_t\}$, such that $\{v_i, \ldots, v_t\}$ is a set of this solution. We then modify this solution by adding node v_{t+1} in the last set, and we obtain an optimal solution for the (k', D)-family-matching problem, for the sub-path induced by the set of nodes $\{v_1, \ldots, v_{t+1}\}$, such that $\{v_i, \ldots, v_{t+1}\}$ is a set of this solution. □

Claim 11. *For every $k' \in \{1, \ldots, k\}$, then*

$$\Psi_{k',D}(v_{t+1}, t+1) = \max_{i \in \{\max(1, t-D), \ldots, t+1\}} \Psi_{k'-1,D}(v_t, i).$$

Furthermore,

$$\Psi_{0,D}(v_{t+1}, t+1) = -\infty.$$

Proof of Claim 11. The set $\{v_{t+1}\}$ must be a set of the solution. Thus, we have to consider an optimal solution for the $(k'-1, D)$-family-matching problem, for the sub-path induced by the set of nodes $\{v_1, \ldots, v_t\}$, such that either node v_t does not belong to any set or $\{v_i, \ldots, v_t\}$ is a set of this solution for some $i \in \{\max(1, t-D), \ldots, t\}$. Indeed, the number of sets of this former solution must be $k'-1$ because $\{v_{t+1}\}$ is a set of an optimal solution for the (k', D)-family-matching problem for the sub-path induced by the set of nodes $\{v_1, \ldots, v_{t+1}\}$. Finally, the second equation is obvious because $\{v_{t+1}\}$ is a set of an optimal solution but the number of sets is zero. □

Claim 12. *For every $k' \in \{0, \ldots, k\}$, then*

$$\Psi_{k',D}(v_{t+1}, t+2) = \max_{i \in \{\max(1, t-D), \ldots, t+1\}} \Psi_{k',D}(v_t, i).$$

Proof of Claim 12. First, node v_{t+1} does not belong to any set of the solution. Thus, we have to consider an optimal solution for the (k', D)-family-matching problem, for the sub-path induced by the set of nodes $\{v_1, \ldots, v_t\}$, such that either node v_t does not belong to any set or $\{v_i, \ldots, v_t\}$ is a set of this solution for some $i \in \{\max(1, t-D), \ldots, t\}$. □

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 $O(kD)$. We get that the time complexity of the dynamic programming algorithm is $O(nkD)$.

To conclude the proof of Lemma 14, when we have computed $\Psi_{k',D}(v_n, i)$ for every $k' \in \{0, \ldots, k\}$ and every $i \in \{\max(1, n-D), \ldots, n+1\}$, then we can deduce an optimal solution $\mathcal{S}$ and the optimal value of the (k, D)-family-matching problem for G. Indeed,

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

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

D.4 The D-family-matching problem for paths

In this section, we illustrate the result of Lemma 6 by considering paths. Consider an intersection path $G = (V, E, w)$. By Lemma 13, 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 15 a better time complexity algorithm for the D-family-matching problem. Indeed, the time complexity is $O(Dn)$.

Lemma. 15 (Computation of $\Phi_D(G)$ for paths)**.** *Let $D \geq 1$ be a positive integer. 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 15. 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 13. *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 13. 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 14.

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

Proof of Claim 14. 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 14, 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. □

D.5 The (k,D)-family-matching problem for cycles

We now prove in Lemma 16 a polynomial time algorithm for the (k, D)-family-matching problem when G is an even cycle. If G is an odd cycle, the time complexity is the same but we only consider here bipartite graphs because the intersection graph of any instance of the (k, D)-family-matching problem is bipartite.

Lemma. 16 (Computation of $\Phi_{k,D}(G)$ for cycles)**.** *Let $k, D \geq 1$ be two positive integers. Consider any intersection graph $G = (V, E, w)$ that is an even cycle. Then, there exists an $O(kD^2n)$-time complexity algorithm for the (k, D)-family-matching problem for G.*

Proof of Lemma 16. The proof is similar to the proof of Lemma 14. We choose any node $v \in V$ and we compute the set $H(G, v)$ of all the non-empty sub-graphs of G that contain v and with diameter at most D. Observe that $|H(G, v)| = O(D)$ because G is a cycle. For every such sub-graph $H \in H(G, v)$, we use the dynamic programming algorithm designed in the proof of Lemma 14 in order to compute an optimal solution for the $(k-1, D)$-family-matching problem for $G_H = (V_H, E_H, w)$, where $V_H = V \setminus V(H)$ and $E_H = E \cap (V_H \times V_H)$. Indeed, G_H is necessarily a path because H is non-empty. Intuitively, for every possible set that contain v, we compute an optimal solution for the graph minus this set (sub-graph). We finally compute

$$\Phi_{k,D}(G) = \max_{H \in H(G,v)} (\Psi_{k-1,D}(G_H) + \sum_{e \in E_H} w_e).$$

We get an $O(kD^2n)$-time complexity algorithm for the (k, D)-family-matching problem for cycles. □

Note that the algorithms described in the proof of Lemma 14 and in the proof of Lemma 16 allow us to compute optimal solutions for all $k \in \{0, \ldots, n\}$.

D.6 The D-family-matching problem for cycles

We now deduce in Corollary 3 an efficient algorithm for the D-family-matching problem when G is an even cycle.

Corollary. 3 (Computation of $\Phi_D(G)$ for cycles)**.** *Let $D \geq 1$ be a positive integer. Consider any intersection graph $G = (V, E, w)$ that is an even cycle. Then, there exists an $O(D^2n)$-time complexity algorithm for the D-family-matching problem for G.*

Indeed, we have

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

D.7 The (k,D)-family-matching problem for union of graphs

We prove in Lemma 17 an exact dynamic programming algorithm for the (k, D)-family-matching problem when the graph G is a disjoint union of graphs.

Lemma. 17 (Computation of $\Phi_{k,D}(G)$ for union of graphs)**.** *Let $k, D \geq 1$ be two positive integers. Consider any intersection graph $G = (V, E, w)$. Let $r \geq 1$ be the number of maximal components of G. Assume that the (k, D)-family-matching problem can be solved in $O(C)$-time for every maximal component of G. Then, there is an $O(|cc(G)|C)$-time complexity algorithm for the (k, D)-family-matching problem for G.*

Proof of Lemma 17. We denote by $\{Q_1, \ldots, Q_\rho\}$ the set of $\rho = |cc(G)| \geq 1$ maximal connected components of G. By definition of the (k, D)-family-matching problem (with diameter constraint), any admissible family $\mathcal{S} = \{S_1, \ldots, S_k\}$ of k disjoint subsets of V is such that for every $i \in \{1, \ldots, k\}$, then there exists $j \in \{1, \ldots, \rho\}$ such that $S_j \subseteq V(Q_j)$. (Otherwise, we would have infinite diameter.)

We now describe a polynomial time dynamic programming algorithm for the (k, D)-family-matching problem. Given $\rho' \in \{1, \ldots, \rho-1\}$, suppose we have computed optimal solutions for the (k', D)-family-matching problem for the graph induced by the set of nodes $\{V(Q_1), \ldots, V(Q_{\rho'})\}$ for every $k' \in \{1, \ldots, k\}$. For $\rho' = 1$, such solutions can be computed (by assumption) with an $O(C)$-time complexity algorithm. Then, we compute optimal solutions for the (k', D)-family-matching problem for $\{Q_1, \ldots, Q_{\rho'}, Q_{\rho'+1}\}$ for every $k' \in \{1, \ldots, k\}$. Let $k' \in \{0, \ldots, k\}$. We compute an optimal solution $S_{\rho'+1,k'}$ for the (k', D)-family-matching problem for $\{Q_1, \ldots, Q_{\rho'}, Q_{\rho'+1}\}$ with k' sets. Let $S_{\rho',j}$ be an optimal solution for the (j, D)-family-matching problem for $\{Q_1, \ldots, Q_{\rho'}\}$ for every $j \in \{0, \ldots, k'\}$. Observe that the number of sets k'_1 of $S_{\rho'+1,k'}$ that are contained in the set of nodes $V(Q_1) \cup \ldots \cup V(Q_{\rho'})$ is such that $k' = k'_1 + k'_2$, where k'_2 is the number of sets of $S_{\rho'+1,k'}$ that are contained in the set of nodes $V(Q_{\rho'+1})$. For every $k'_1 \in \{0, \ldots, k'$, we compute an optimal solution $S'_{k'_1}$ for the $(k' - k'_1, D)$-family-matching problem for $Q_{\rho'+1}$. Note that the algorithms designed in this paper can return (in general) all the solutions (that is for all possible values of $(k' - k'_1)$ after one execution). Then, we compute an optimal solution by choosing k'_1 that maximizes the sum $\Phi(S_{\rho',k'_1}) + \Phi(S'_{k'_1})$. We do that computation for every $k' \in \{0, \ldots, k\}$. Thus, we get the optimal solutions for the (k', D)-family-matching problem for $\{Q_1, \ldots, Q_{\rho'+1}\}$ for every $k' \in \{1, \ldots, k\}$. We do this until having optimal solutions for the (k', D)-family-matching problem for $\{Q_1, \ldots, Q_\rho\}$ and for every $k' \in \{1, \ldots, k\}$. We finally return the optimal solution with k sets.

To conclude the proof of Lemma 17, we prove that the time complexity is $O(\rho C)$. Without loss of generality, let Q_1 be the largest (in terms of number of nodes) maximal connected component of G. By assumption, the time complexity of computing a solution for the (k, D)-family-matching problem for Q_i is $O(C)$ for every $i \in \{1, \ldots, \rho\}$. We get $O(\rho C)$ for such a computation for all components.

We now show the time complexity of the dynamic programming algorithm previously explained. For every $\rho' \in \{1, \ldots, \rho-1\}$, the time complexity for computing $S_{\rho'+1,k'}$ is $O(\min(|V(Q_1)|, k'))$ for any $k' \in \{0, \ldots, k\}$. Indeed, there are at most $\min(|V(Q_1)|, k')$ different solutions for $Q_{\rho'+1}$ (recall that $|V(Q_{\rho'+1})| \leq |V(Q_1)|$) because the number of different solutions returned by the algorithm for the (k, D)-family-matchingfor $Q_{\rho'+1}$ is at most k and at most the number of nodes of $|V(Q_{\rho'+1})|$. (the number of sets cannot larger than the number of nodes) We do that computation for every $k' \in \{0, \ldots, k\}$ and every $\rho' \in \{1, \ldots, \rho\}$. We get that the time complexity of the dynamic programming algorithm is $O(\rho k \min(|V(Q_1)|, k))$. Thus, the total complexity is $O(\rho C + \rho k \min(|V(Q_1)|, k))$. Since $C = \Omega(k \min(|V(Q_1)|, k))$, we get that the global time complexity is $O(\rho C)$. Indeed, any algorithm for the (k, D)-family-matching problem will depend

(at least) linearly on the number of nodes and (at least linearly) on the (maximum) number of sets (recall that we compute the solutions for all possible number of sets of size at most k). □

Since $|cc(G)||V(Q_j)| = O(n)$ for every $j \in \{1, \ldots, |cc(G)|\}$, we deduce in Corollary 4 dynamic programming algorithms for the (k, D)-family-matching problem for some classes of graphs.

Corollary. 4. *Let $k, D \geq 1$ be two positive integers. Let $G = (V, E, w)$ be an intersection graph. Then, there exists a dynamic programming algorithm for the (k, D)-family-matching problem for G with time complexity*

- *$O((2k)^{\Delta} D^2 \Delta^2 n)$ when G is a union of trees;*
- *$O(kDn)$ when G is a union of paths;*
- *$O(kD^2 n)$ when G is a graph of maximum degree two (union of paths and cycles).*

We now address our result in terms of the original problem (Corollary 5).

Corollary. 5. *Let $k, D \geq 1$ be two positive integers. Consider any instance of the (k, 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 $O((r+r')kD^2)$-time complexity algorithm for the (k, D)-family-matching problem.

Say otherwise, Corollary 5 shows that there is a polynomial time algorithm for the (k, D)-family-matching problem if any set in $F \cup F'$ has a non-empty intersection with at most two other sets of $F \cup F'$.

D.8 The D-family-matching problem for union of graphs

We prove in Lemma 18 an exact dynamic programming algorithm for the D-family-matching problem when the graph G is a disjoint union of graphs.

Lemma. 18 (Computation of $\Phi_D(G)$ for union of graphs)**.** *Let $D \geq 1$ be a positive integer. Consider any intersection graph $G = (V, E, w)$. Let $r \geq 1$ be the number of maximal components of G. Assume that the D-family-matching problem can be solved in $O(C)$-time for every maximal component of G. Then, there exists an $O(|cc(G)|C)$-time complexity algorithm for the D-family-matching problem for G.*

Proof of Lemma 18. We denote by $\{Q_1, \ldots, Q_\rho\}$ the set of $|cc(G)| \geq 1$ maximal connected components of G. By definition of the D-family-matching problem, we get that $\Phi_D(G) = \sum_{j=1}^{\rho} \Phi_D(Q_j)$. The result directly follows. □

Since $|cc(G)||V(Q_j)| = O(n)$ for every $j \in \{1, \ldots, |cc(G)|\}$, we deduce in Corollary 6 polynomial time dynamic programming algorithms for the D-family-matching problem for some classes of graphs.

Corollary. 6. *Let $D \geq 1$. Let $G = (V, E, w)$ be an intersection graph. Then, there exists a dynamic programming algorithm for the D-family-matching problem for G with time complexity*

- *$O(D^2\Delta^2 n)$ when G is a union of trees;*
- *$O(Dn)$ when G is a union of paths;*
- *$O(D^2 n)$ when G is a graph of maximum degree two (union of paths and cycles).*

We now address our result in terms of the original problem (Corollary 7).

Corollary. 7. *Let $D \geq 1$. 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 7 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 $F \cup F'$.

E Appendix - Efficient algorithms for general graphs

E.1 Warm up

In this section, we design several algorithms parametrized by the maximum degree and the diameter of the graphs. Say otherwise, the time complexity of each of them is a function of the maximum degree and the diameter of the graph. We focus on the (k, D)-family-matching problem that is easier than the D-family-matching problem in terms of techniques developed in this section. Observe that it is the contrary in Section 4.

Lemma. 19. *Let $k, D \geq 1$ be two positive integers. Consider any intersection graph $G = (V, E, w)$. Then, there exists an $O(n^k h(G)^k)$-time complexity algorithm for the (k, D)-family-matching problem for G.*

Proof of Lemma 19. The algorithm consists in enumerating all the possible sets of k sub-graphs of $H(G)$. The number of such sets is

$$\binom{h(G)n}{k}$$

and so the time complexity of the algorithm is $O(n^k h(G)^k)$. □

Lemma. 20. *Let $k, D \geq 1$ be two positive integers. Consider any intersection graph $G = (V, E, w)$. Then, there exists an $O(n^k 2^{k\Delta(1-\Delta^D)(1-\Delta)^{-1}})$-time complexity algorithm for the (k, D)-family-matching problem for G.*

Proof of Lemma 20. Let $v \in V$ be any node of G. We first prove an upper-bound on $h(G, v)$. We enumerate all possible sub-graphs $H \in H(G, v)$ as follows. There are at most 2^{Δ} possible different neighbors of v in H. There are at most 2^{Δ^2} possible different nodes that are at distance 2 of v in H. More generally, there are at most 2^{Δ^i} possible different nodes that are at distance i of v in H, for every i, $1 \leq i \leq D$. It follows that

$$h(G, v) = \prod_{i=1}^{D} 2^{\Delta^i} = 2^{\sum_{i=1}^{D} \Delta^i} = 2^{\Delta(1-\Delta^D)(1-\Delta)^{-1}}.$$

We deduce that

$$h(G) = O(2^{\Delta(1-\Delta^D)(1-\Delta)^{-1}}).$$

The algorithm consists in enumerating all the possible sets of k sub-graphs of $H(G)$. The number of such sets is

$$\binom{h(G)n}{k} = \binom{2^{\Delta(1-\Delta^D)(1-\Delta)^{-1}}n}{k}$$

and so the time complexity of the algorithm is $O(n^k 2^{k\Delta(1-\Delta^D)(1-\Delta)^{-1}})$. □

Corollary. 8. *Let $k, D \geq 1$ be two positive integers. Consider any intersection graph $G = (V, E, w)$. If k, Δ, and D are constant numbers, then there exists a polynomial time algorithm for the (k, D)-family-matching problem for G.*

Corollary. 9. *Let $k, D \geq 1$ be any two constant integers. Let $\beta \geq 1$ be any constant integer. Consider any instance of the (k, D)-family-matching problem such that $|F_i| \leq \beta$ and $|F'_j| \leq \beta$ for every i, j, $1 \leq i \leq n$, $1 \leq j \leq n'$. Then, there exists a polynomial time algorithm for the (k, D)-family-matching problem for G.*

We prove another algorithm by using solutions of the (Degree, Diameter) problem, also called (Δ,D) problem. Given $\Delta \geq 1$ and $D \geq 1$, a (Δ,D)-graph is a graph with maximum degree Δ and diameter at most D. The maximum number of nodes of a (Δ,D)-graph is denoted $N(\Delta, D)$. Such a graph (and the associated number of nodes) is an optimal solution of the (Δ,D) problem.

Lemma. 21. *Let $k, D \geq 1$ be two positive integers. Consider any intersection graph $G = (V, E, w)$. Then, there exists an $O(n^{k.N(\Delta,D)})$-time complexity algorithm for the (k, D)-family-matching problem for G.*

Proof of Lemma 21. We first prove that $nh(G) = O(\binom{n}{N(\Delta,D)})$. Indeed, by definition of $N(\Delta, D)$, and so of the (Δ,D) problem, there is no sub-graph of G with diameter at most D and with maximum degree Δ (that is the maximum degree of G) such that the number of nodes is at least $N(\Delta, D) + 1$. The algorithm consists in enumerating all the possible sets of k sub-graphs. The number of such sets is

$$\binom{\binom{n}{N(\Delta,D)}}{k}$$

and so the time complexity is $O(n^{k.N(\Delta,D)})$. □

The Moore bound shows that $N(\Delta, D) \leq (\Delta(\Delta-1)^D - 2)(\Delta-2)^{-1}$ [2]. We deduce Corollary 10.

Corollary. 10. *Let $k, D \geq 1$ be two positive integers. Consider any intersection graph $G = (V, E, w)$. Then, there exists an $O(n^{k(\Delta(\Delta-1)^D-2)(\Delta-2)^{-1}})$-time complexity algorithm for the (k, D)-family-matching problem for G.*

E.2 Dynamic programming algorithms under spanning tree constraint

We define the (k, D)-family-matching problem for G constrained by T_r, that consists in only considering sets that belong to $H(G, T_r)$.

Definition. 11 ((k,D)-family-matching constrained by a tree)**.** *Let $k, D \geq 1$ be two positive integers. Let $G = (V, E, w)$ be an intersection graph and let T_r be a spanning tree of G rooted at $r \in V$. A (k, D)-family-matching for G constrained by T_r is a (k, D)-family-matching $\mathcal{S}$ for G such that for every $S \in \mathcal{S}$, then there exists $H \in H(G, T_r)$ such that $S = H$.*

The set $\mathcal{S}_{k,D}(G,T_r)$ is the set of all the (k,D)-family-matching constrained by T_r.

Definition. 12 ((k,D)-family-matching problem constrained by a tree)**.** *Let $k, D \geq 1$ be two positive integers. Given an intersection graph G and a rooted spanning tree T_r of G, the (k,D)-family-matching problem consists in computing*

$$\Phi_{k,D}(G,T_r) = \max_{\mathcal{S}\in\mathcal{S}_{k,D}(G,T_r)} \Phi(\mathcal{S}).$$

Lemma. 22 (Computation of $\Phi_{k,D}(G,T_r)$)**.** *Let $k, D \geq 1$ be two positive integers. Let $G = (V,E,w)$ be any intersection graph and let T_r be any spanning tree of G rooted at node $r \in V$. Then, there exists an $O(k^{\Delta}h(G,T_r)^{\Delta}n)$-time complexity algorithm for the (k,D)-family-matching problem for G constrained by T_r.*

Proof of Lemma 22. 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 $\Psi_{k',D}$ as follows. For every $v \in V$, every $k' \in \{0,\dots,k\}$, and every $H \in H(G,T_r,v)$, then $\Psi_{k',D}(v,H)$ is the score of an optimal solution $\mathcal{S}$ for the (k',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 $\Psi_{k',D}(v,H) = -\infty$. This is the case when, for instance, $|V(T_v)| < k'$.

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

- $\Psi_{0,D}(v,H) = 0$ if $H = (\emptyset,\emptyset)$,
- $\Psi_{0,D}(v,H) = -\infty$ if $H \neq (\emptyset,\emptyset)$,
- $\Psi_{1,D}(v,H) = 0$ if $H = (\{v\},\emptyset)$,
- $\Psi_{1,D}(v,H) = -\infty$ if $H \neq (\{v\},\emptyset)$,
- $\Psi_{k',D}(v,i) = -\infty$ for every $k' \in \{2,\dots,k\}$ and every $H \in H(G,T_r,v)$.

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 $N(v) = \{v_1,\dots,v_q\}$ be the set of $q \geq 1$ neighbors of v in T_v. Suppose we have computed $\Psi_{k',D}(v_j,H)$ for every $j \in \{1,\dots,q\}$, every $k' \in \{0,\dots,k\}$, and every $H \in H(G,T_r,v_j)$. We prove that we can compute $\Psi_{k',D}(v,H)$ for every $k' \in \{0,\dots,k\}$ and every $H \in H(G,v)$. There are three different cases (corresponding to the three following claims).

Claim 15. *For every $k' \in \{0,\dots,k\}$, then*

$$\Psi_{k',D}(v,(\emptyset,\emptyset)) = \max_{(k_1,\dots,k_q)\in\mathcal{K}} \Big(\max_{(H'_1,\dots,H'_q)\in\mathcal{H}'} \sum_{j=1}^{q} \Psi_{k_j,D}(v_j,H'_j)\Big),$$

where $\mathcal{H}'$ is the set of all vectors $(H'_1,\dots,H'_q)$ such that $H'_i \in H'(G,T_r,v_i)$ for every $i \in \{1,\dots,q\}$, and where $\mathcal{K}$ is the set of all vectors $(k_1,\dots,k_q)$ such that

- *$0 \leq k_j \leq k'$ for every $j \in \{1,\dots,q\}$,*
- *$k_j \leq |T_{v_j}|$ for every $j \in \{1,\dots,q\}$,*
- *$\sum_{j=1}^{q} k_j = k'$.*

Proof of Claim 15. We consider here an optimal solution for the (k', 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, $\Psi_{k',D}(v, (\emptyset, \emptyset))$ consists in chosing the number of sets k_j that belong to T_{v_j} and the set H'_j that contains v_j (possibly empty) for every $j \in \{1, \ldots, q\}$, such that the score is maximal. Note that $\sum_{j=1}^{q} k_j = k'$. These choices correspond to the two maximum functions. □

Claim 16. *For every $k' \in \{0, \ldots, k\}$, then*

$$\Psi_{k',D}(v, (\{v\}, \emptyset)) = \max_{(k_1,\ldots,k_q)\in\mathcal{K}} \Big(\max_{(H'_1,\ldots,H'_q)\in\mathcal{H}'} \sum_{j=1}^{q} \Psi_{k_j,D}(v_j, H'_j)\Big),$$

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 $i \in \{1, \ldots, q\}$, and where $\mathcal{K}$ is the set of all vectors $(k_1, \ldots, k_q)$ such that

- $0 \leq k_j \leq k'$ *for every* $j \in \{1, \ldots, q\}$,
- $k_j \leq |T_{v_j}|$ *for every* $j \in \{1, \ldots, q\}$,
- $\sum_{j=1}^{q} k_j = k' - 1$.

Proof of Claim 16. This proof is similar to the proof of Claim 15. Indeed, we consider an optimal solution $\mathcal{S}$ for the (k', 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, $\Psi_{k',D}(v, (\{v\}, \emptyset))$ consists in chosing the number of sets k_j that belong to T_{v_j} and the set H'_j that contains v_j (possibly empty) for every $j \in \{1, \ldots, q\}$, such that the score is maximal. Note that $\sum_{j=1}^{q} k_j = k' - 1$ because $\{v\}$ is already in the solution. These choices correspond to the two maximum functions. □

Claim 17. *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 $V(H_{v_j}) = V(H) \cap V(T_{v_j})$. For every $k' \in \{0, \ldots, k\}$, then*

$$\Psi_{k',D}(v, H) = \sum_{e'\in E(H_v)} w_{e'} \; +$$

$$\max_{(k_1,\ldots,k_q)\in\mathcal{K}} \Big(\sum_{j=1}^{q'} (\Psi_{k_j,D}(v_j, H_{v_j}) - (\sum_{e'\in E(H_{v_j})} w_{e'}) + \max_{(H'_{q'+1},\ldots,H'_q)\in\mathcal{H}'} \sum_{j=q'+1}^{q} \Psi_{k_j,D}(v_j, H'_j))\Big),$$

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 $i \in \{q'+1, \ldots, q\}$, and where $\mathcal{K}$ is the set of all vectors $(k_1, \ldots, k_q)$ such that

- $0 \leq k_j \leq k'$ *for every* $j \in \{1, \ldots, q\}$,
- $k_j \leq |T_{v_j}|$ *for every* $j \in \{1, \ldots, q\}$,
- $\sum_{j=1}^{q} k_j = k' - q' - 1$.

Proof of Claim 17. We consider an optimal solution $\mathcal{S}$ for the (k', 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, $\Psi_{k',D}(v, H)$ consists, for every $j \in \{1, \ldots, q'\}$, in choosing the number of

sets k_j that belong to T_{v_j} (one of them will be merged in the set containing v) and, for every $j \in \{q'+1, \ldots, q\}$, consists in choosing the number of sets k_j that belong to T_{v_j} and 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. Note also that $\sum_{j=1}^{q'} k_j = k' + q' - 1$ because q' chosen sets will be merged with the set containing v. □

For every $v \in V$, we address the time complexity of computing Ψ as follows. The time complexity of the computation done in Claim 15 is $O(k^{q+1} \prod_{j=1}^{q} h(G, T_r, v_i))$. The time complexity of the computation done in Claim 16 is $O(k^{q+1} \prod_{j=1}^{q} h'(G, T_r, v_i))$. The time complexity of the computation done in Claim 17 is $O(h(G, T_r, v) k^{q+1} (q' + |E(H_{v_j})| + \prod_{j=q'+1}^{q} h(G, T_r, v_i)))$. 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 $O(k^{\Delta} h(G, T_r)^{\Delta} n)$.

To conclude the proof of Lemma 22, when we have computed $\Psi_{k',D}(r, H)$ for every $H \in H(G, T_r, r)$ and every $k' \in \{0, \ldots, k\}$, we can deduce an optimal solution $\mathcal{S}$ and the optimal value of the (k, D)-family-matching problem for G constrained by T. Indeed,

$$\Phi_{k,D}(G, T_r) = \max_{H \in H(G, T_r, r)} \Psi_{k,D}(r, H).$$

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

Proof of Lemma 7. 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). The proofs of the claims are similar to the ones of Lemma 22.

Claim 18.

$$\Psi_D(v, (\emptyset, \emptyset)) = \max_{(H'_1, \ldots, H'_q) \in \mathcal{H}'} \sum_{j=1}^{q} \Psi_D(v_j, H'_j),$$

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 $i \in \{1, \ldots, q\}$.

Claim 19.

$$\Psi_D(v, (\{v\}, \emptyset)) = \max_{(H'_1, \ldots, H'_q) \in \mathcal{H}'} \sum_{j=1}^{q} \Psi_D(v_j, H'_j),$$

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 $i \in \{1, \ldots, q\}$.

Claim 20. *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 $i \in \{q'+1, \ldots, q\}$.

For every $v \in V$, we address the time complexity of computing Ψ as follows. The time complexity of the computation done in Claim 18 is $O(\prod_{j=1}^{q} h(G, T_r, v_i))$. The time complexity of the computation done in Claim 19 is $O(\prod_{j=1}^{q} h'(G, T_r, v_i))$. The time complexity of the computation done in Claim 20 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 $O(h(G, T_r)^{\Delta} n)$.

To conclude the proof of Lemma 7, 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 $\mathcal{S}$). □

Corollary 11 shows that if D and Δ are constant integers, then the algorithms described in Lemma 22 and in Lemma 7 have polynomial time complexity. Indeed, in that case, $h(G, T) = O(2^{\Delta^D})$.

Corollary. 11. *Let $k \geq 1$ be a positive integer. Let $D \geq 1$ be any constant positive integer. Consider any intersection graph $G = (V, E, w)$ of bounded maximum degree $\Delta = O(1)$. Let T be any spanning tree of G rooted at node $r \in V$. Then, there exist polynomial time algorithms*

- *for the (k, D)-family-matching problem for G constrained by T*
- *and for the D-family-matching problem for G constrained by T.*

But, even if D is not a constant, we obtain in Corollary 12 a polynomial time algorithm if $h'(G)$ is a polynomial in n. However, we stil need $\Delta = O(1)$.

Corollary. 12. *Let $k, D \geq 1$ be two positive integers. Consider any intersection graph $G = (V, E, w)$ of bounded maximum degree $\Delta = O(1)$. Let T be any spanning tree of G rooted at node $r \in V$. If $h(G, T)$ is a polynomial in n, then there exist polynomial time algorithms*

- *for the (k, D)-family-matching problem for G constrained by T*
- *and for the D-family-matching problem for G constrained by T.*

E.3 Algorithms based on spanning trees

Lemma. 23. *Let $k, D \geq 1$ be two positive integers. Consider any intersection graph G. Then, there exists a rooted spanning tree T of G such that $\Phi_{k,D}(G) = \Phi_{k,D}(G, T)$.*

Proof of Lemma 23. Consider an optimal solution $\mathcal{S} = \{S_1, \ldots, S_k\}$ for the (k, 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 (k, D)-family-matching problem for G constrained by T. Thus, $\Phi_{k,D}(G, T) = \Phi_{k,D}(G)$. □

Let $\mathcal{T}(G)$ be the set of all different rooted spanning trees of G. We deduce in Corollary 13 an exact algorithm for the (k, D)-family-matching problem for G.

Corollary. 13. *Given any two positive integers $k, D \geq 1$ and any intersection graph G, Algorithm 2 returns $\Phi_{k,D}(G)$, that is an optimal solution for the (k, D)-family-matching problem for G, if:*

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

Furthermore, the time complexity is of Algorithm 2 is $O(|\mathcal{T}(G)| k^{\Delta} \max_{T_r \in \mathcal{T}(G)} h(G, T_r)^{\Delta} n)$.

Proof of Lemma 8. 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, $\Phi_D(G, T) = \Phi_D(G)$. □

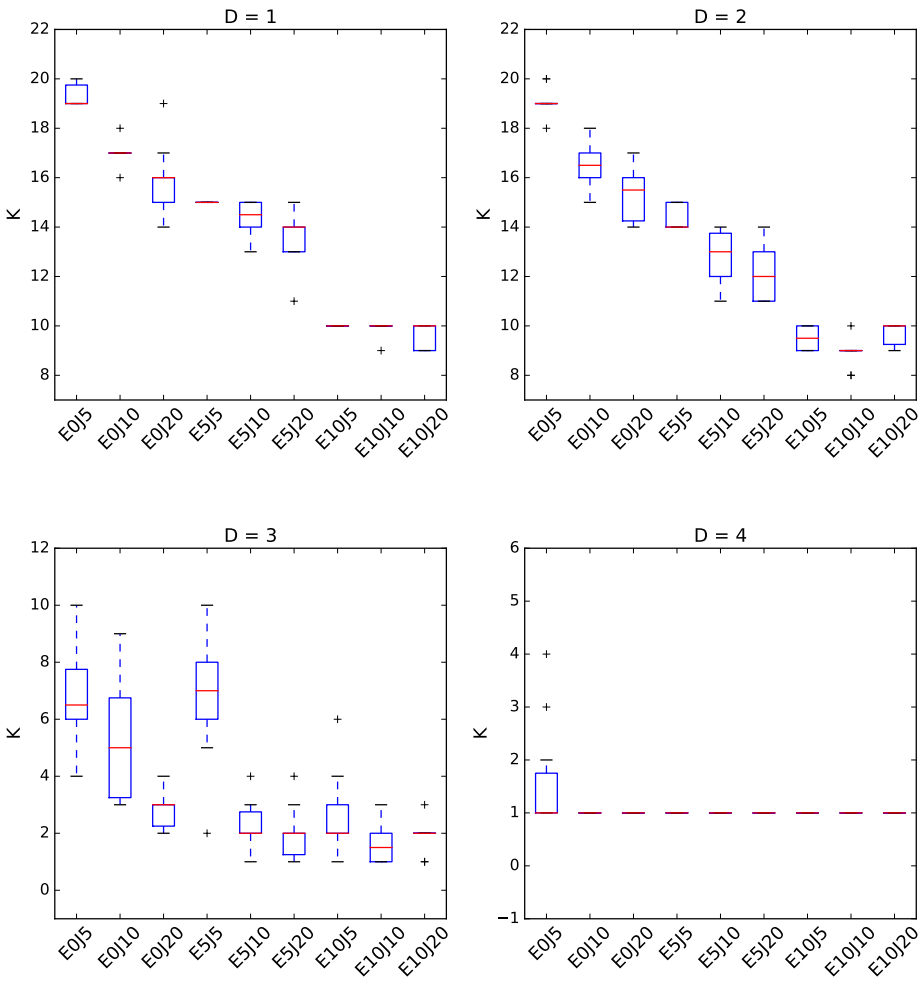
Algorithm 2 Generic algorithm for the (k, D)-family-matching problem.

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

```
1: M := ∅, t := 0
2: while ¬ Π(M) do
3:     t := t + 1
4:     Compute the spanning tree T^t := R(G, t)
5:     Compute S^t by using Algorithm A(G, T^t, k, D)
6:     M := M ∪ S^t
7: return S ∈ M of maximum score
```

F Appendix - Experiments

Figure 7 Algorithm $RST(G, D)$ for clustering with $(t = 1000, r = 20)$. Run for $n_i = 10000$ iterations. Best value for k as a function of the 9 scenarii.



Results

Figure 8 Algorithm $RST(G, D)$ for clustering with $(t = 1000, r = 20)$. Run for $n_i = 10000$ iterations. Score Φ as a function of the 9 scenarii.

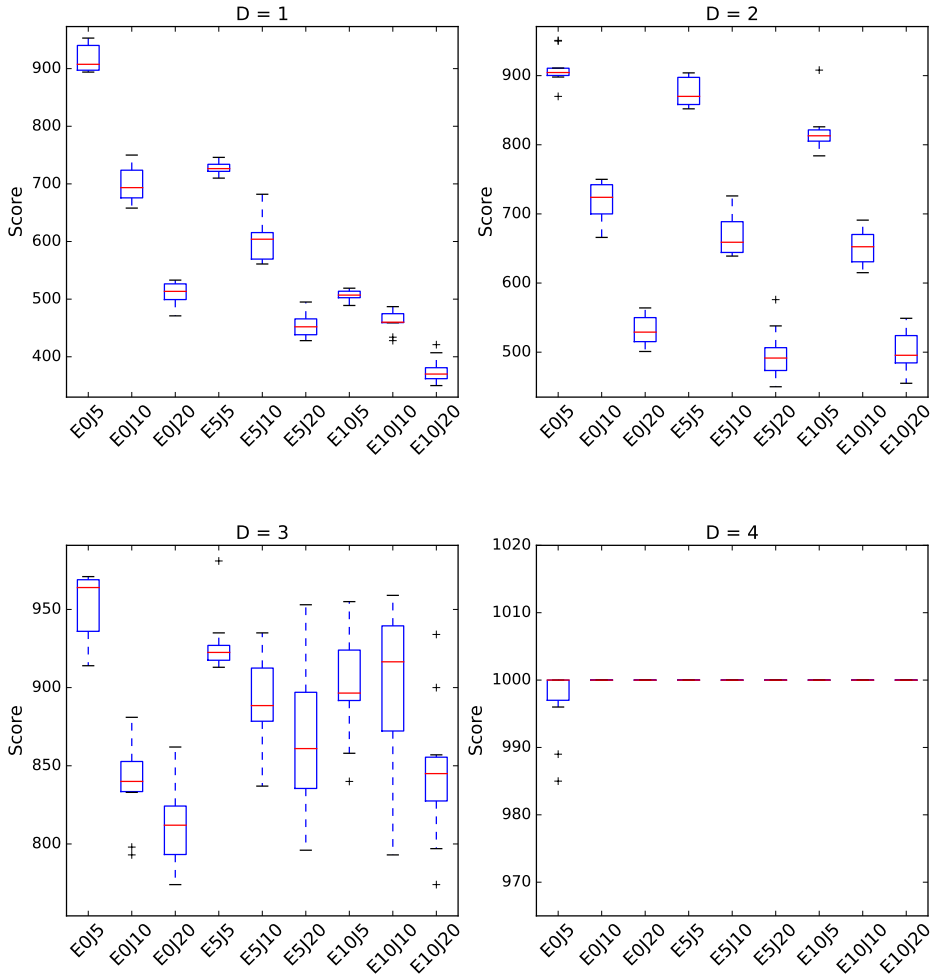


Figure 9 Algorithm $RST(G, D)$ for clustering with $(t = 3000, r = 50)$. Run for $n_i = 10000$ iterations. Best value for k as a function of the 9 scenarii.

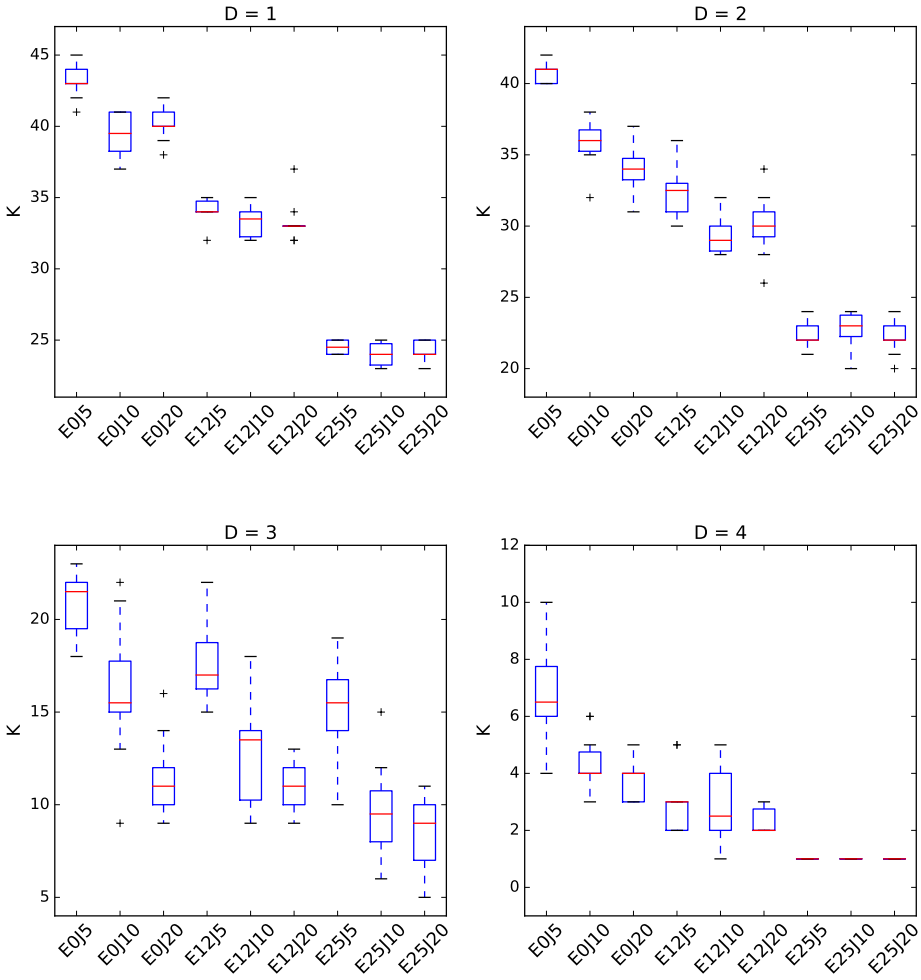


Figure 10 Algorithm $RST(G, D)$ for clustering with $(t = 3000, r = 50)$. Run for $n_i = 10000$ iterations. Score Φ as a function of the 9 scenarii.

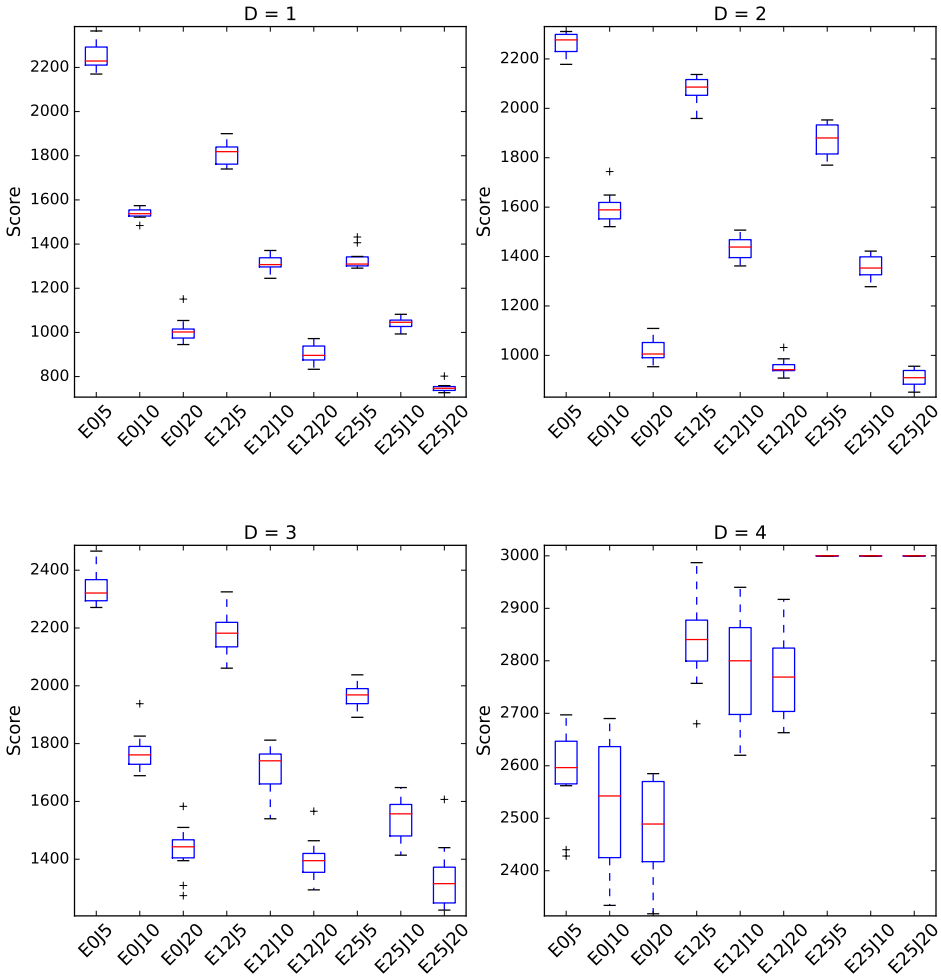


Figure 11 Algorithm $MST(G, D)$ for clusterings with $(t = 3000, r = 50)$. Best value for k as a function of the 9 scenarii.

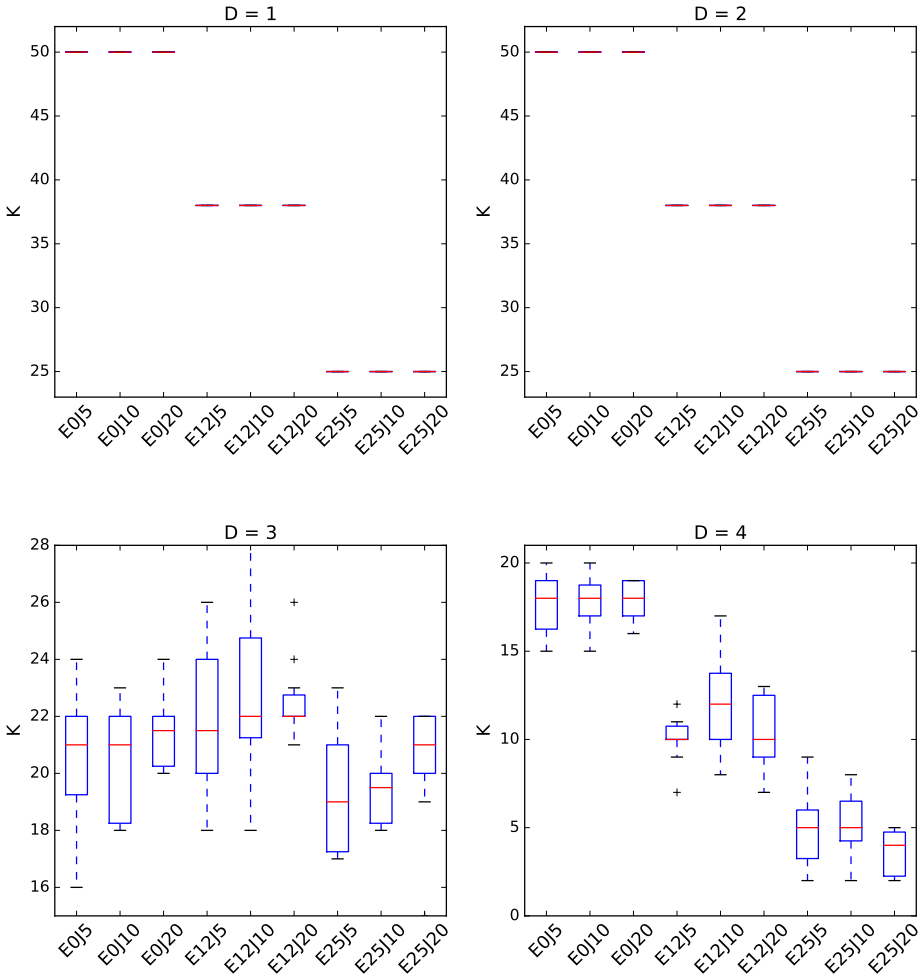
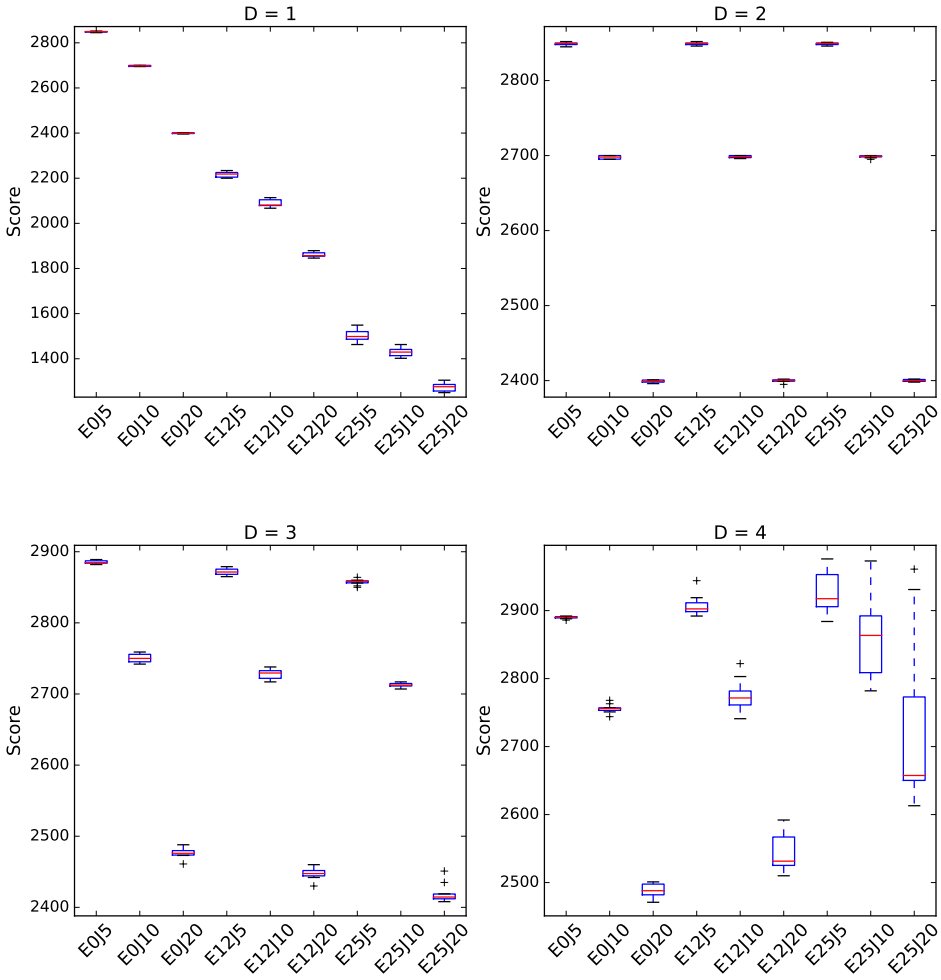


Figure 12 Algorithm $MST(G, D)$ for clusterings with $(t = 3000, r = 50)$. Scores Φ as a function of the 9 scenarii.



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