Efficient Approximation of Probability Distributions with k-order Decomposable Models

Aritz Pérez[a], Iñaki Inza[b], Jose A. Lozano[b,a]

[a]*Basque Center for Applied Mathematics (BCAM), Bilbao, Spain*
[b]*Department of Computer Science and Artificial Intelligence, University of the Basque Country, San Sebastián, Spain*

Abstract

During the last decades several learning algorithms have been proposed to learn probability distributions based on decomposable models. Some of these algorithms can be used to search for a maximum likelihood decomposable model with a given maximum clique size, k. Unfortunately, the problem of learning a maximum likelihood decomposable model given a maximum clique size is NP-hard for $k > 2$. In this work, we propose the **fractal tree** family of algorithms which approximates this problem with a computational complexity of $\mathcal{O}(k^2 \cdot n^2 \cdot N)$ in the worst case, where n is the number of implied random variables and N is the size of the training set.

The fractal tree algorithms construct a sequence of maximal i-order decomposable graphs, for $i = 2, ..., k$, in $k - 1$ steps. At each step, the algorithms follow a divide-and-conquer strategy that decomposes the problem into a set of separator problems. Each separator problem is efficiently solved using the generalized Chow-Liu algorithm. Fractal trees can be considered a natural extension of the Chow-Liu algorithm, from $k = 2$ to arbitrary values of k, and they have shown a competitive behavior to deal with the maximum likelihood problem. Due to their competitive behavior, their low computational complexity and their modularity, which allow them to implement different parallelization strategies, the proposed procedures are especially advisable for modeling high dimensional domains.

Keywords: Approximating probability distributions, learning decomposable models, bounded clique size, maximum likelihood problem, the Chow-Liu algorithm.

1. Introduction

In order to deal with some Statistical and Artificial Intelligence problems, a probability distribution is required. In many of these problems the probability distribution is not explicitly given and only a set of independent samples, distributed according to it, is available. When a sufficiently large set of samples is accessible, we can approximate the probability distribution using the empirical joint distribution. However, the number of samples required to obtain a reliable estimation of the joint distribution grows exponentially with the number of the implied random variables. In order to learn robust probabilistic models from the available data, the joint distribution is approximated using a product of functions with a smaller number of parameters, e.g., marginal probability distributions. These models are usually learned from data by means of a learning algorithm. From a practical point of view, it is desirable to develop learning algorithms with a low computational complexity in order to deal with high dimensional domains. In addition, the algorithms should learn probabilistic models with a low tree-width that allow to perform

probabilistic inference tasks efficiently. During the last decades, probabilistic graphical models have provided one of the most effective tools for the automatic learning of probabilistic models (Pearl, 1988), the family of decomposable models being one of the most attractive due to their advantageous theoretical properties (Lauritzen, 1996). In this work, we propose a set of efficient algorithms for learning decomposable models.

The approaches for learning factorized models of the joint distribution from data can be divided into qualitative and quantitative algorithms. The qualitative approaches guide the search of the structure of the factorization using (conditional) independence testing procedures, while the quantitative approaches try to maximize a score related to the goodness of the approximation. Quantitative approaches are usually based on decomposable scores such as log likelihood, Bayesian Dirichlet equivalent metric, minimum description length or Bayesian information criterion (Koller and Friedman, 2009), among others. Our contributions consist of quantitative algorithms based on the maximization of the likelihood score.

The likelihood score quantifies the chance of observing a set of samples under the hypothesis that they are distributed according to a given probability model. It can be also interpreted as the degree of fitness of the model to the available data. Likelihood is directly related with the Kullback-Leibler divergence between the factorization and the empirical distribution[1]. It is known that the likelihood of a model tends to be higher as its complexity increases, because the fitting ability tends to increase with the number of free parameters. However, the risk of overfitting increases with the complexity of the model, which can lead to models with a poor generalization capability. Other scores, such as minimum description length, can be seen as a penalized version of the likelihood. The penalized scores can be interpreted as a trade-off between the degree of fitness to the available data and the complexity of the model. Since the likelihood does not penalize the complexity of the model, an explicit control of the complexity of the model is required to avoid the overfitting phenomenon. In the proposed algorithms, the complexity of the learned models is controlled by means of a single regularization parameter, k, which determines the maximum clique size and, hence, the number of parameters of the learned model. Additionally, the regularization parameter effectively controls the computational complexity required to perform inference tasks since this is exponential in the maximum clique size for decomposable models. This work is focused on learning **maximal k-order decomposable graphs** (**MkDG**, see Definition 5), also known as $(k-1)$-hypertrees (Srebro, 2000), because they are the graphical support of the decomposable models that maximize the likelihood with a maximum clique size of k (Malvestuto, 1991).

One of the most popular quantitative approaches based on the likelihood score for learning probability distributions with a low number of parameters is the Chow-Liu algorithm (**CL**) (Chow and Liu, 1968). CL finds a maximum likelihood probability model among the (possible) huge space of n^{n-2} candidate models with a tree structure. Additionally, it has been proven that the algorithm is asymptotically consistent (Chow and Wagner, 1971). An efficient implementation of the algorithm is based on Prim's algorithm with adjacency lists for maximization, where the weights of the edges correspond to the empirical mutual information between the implied random variables. Given the required weights, it has a computational complexity of $\mathcal{O}(n^2)$, where n is the number of implied random variables. Thus, due to its low computational complexity, the huge number of candidate models that can be attained and its optimality, the algorithm is an excellent building block for designing novel search strategies focused on the maximization of the likelihood.

[1] i.e., the higher the likelihood, the lower the Kullback-Leibler divergence

As we noted before, decomposable models are a popular class of probabilistic models due to their theoretical properties. Among these properties, we highlight the closed form of the maximum likelihood parameters, the interpretation of the models in terms of conditional independences using a graphical criterion, and that they are the basis of the most popular inference algorithms over probability distributions (Lauritzen, 1996).

1.1. Related work

During the last decades, many quantitative algorithms have been proposed in order to learn decomposable models (Malvestuto, 1991; Srebro, 2000; Bach and Jordan, 2001; Desphande et al., 2001; Karger and Srebro, 2001; Ding et al., 2007; Srebro, 2003; Chechetka and Guestrin, 2008; Kovács and Szántai, 2010; Szántai and Kovács, 2011; Malvestuto, 2012; Szántai and Kovács, 2012; Corander et al., 2013; Proulx and Zhang, 2014; Kangas et al., 2014). We would like to emphasize the seminal work of Malvestuto (1991), which establishes the basis for learning decomposable models by means of greedy procedures. This work provides much of the theoretical background presented in Section 2, and it demonstrates that the structure that maximizes the likelihood, given a maximum clique size, is an MkDG. In (Desphande et al., 2001), the authors present, based on the results provided by Lauritzen (1996), a formal characterization of the set of edges that can be added to a decomposable graph maintaining its decomposability, and they also design an algorithm for its identification with a computational complexity of $\mathcal{O}(n^2)$. In addition, assuming that all the quantities required for guiding the search have already been computed, they present a forward greedy algorithm that can learn MkDGs with a computational complexity of $\mathcal{O}(k \cdot n^3)$, in the worst case. In Ding et al. (2007), a modification to the greedy procedure proposed in Desphande et al. (2001) is presented. This procedure is related to the FT family of algorithms because it constructs an MkDG by obtaining a sequence of MiDGs for $i = 2, ..., k$. The main difference with respect to FT is that, at each step, the procedure proposed in Ding et al. (2007) adds a single edge to the structure. Once an edge is added, it identifies new candidate edges using the procedure proposed in Desphande et al. (2001), which leads to a computational complexity of $\mathcal{O}(k \cdot n^3)$. As far as we know, these two algorithms are the most efficient approaches for learning MkDGs. In this work, under the same assumption, we present a family of algorithms with a computational complexity of $\mathcal{O}(k \cdot n^2)$ in the worst case. The work of Srebro (2000, 2003) demonstrates that the learning of maximum likelihood decomposable models, with k greater than 2 and lower than $n - 1$, is NP-hard. Recently, two exact algorithms have been proposed in order to deal with this problem (Corander et al., 2013; Kangas et al., 2014). In (Corander et al., 2013) the authors characterize decomposable graphs using the junction tree representation and the balancing condition. Then, they cast this characterization as a constraint satisfaction problem which is formalized in the language of propositional logic. In (Kangas et al., 2014) an alternative characterization of decomposable graphs based on recursive trees is used. This characterization naturally yields a dynamic programming approach. Unfortunately, most of the approximate algorithms proposed to deal with this problem are exponential in k (Chechetka and Guestrin, 2008; Karger and Srebro, 2001; Bach and Jordan, 2001; Kovács and Szántai, 2010; Malvestuto, 1991; Srebro, 2000, 2003; Szántai and Kovács, 2011) and, thus, they are unpractical for dealing with high dimensional probability distributions, even for moderate values of k.

Recently, in (Kovács and Szántai, 2010; Szántai and Kovács, 2012; Proulx and Zhang, 2014), a set of algorithms related to our proposal were presented. In (Kovács and Szántai, 2010), the authors proposed a procedure for learning an M3DG coarser than an M2DG (a tree). This procedure is generalized in (Szántai and Kovács, 2012) to learn an M$(k + 1)$DG coarser than a given MkDG. In (Proulx and Zhang, 2014) the generalization is adapted in order to be guided

by the log likelihood. The algorithms are focused on the cliques rather than on the separators and they are originally proposed to avoid the learning of an M$(k+1)$DG from scratch. They are based on a junction tree representation of the graph which determines a specific neighborhood of the cliques. Depending on the chosen junction tree, different structures can be attained by the procedures. In other words, they do not take into account all the possible edges that can be added to form an M$(k+1)$DG. These algorithms generate cliques of size $k+1$ considering pairs of neighbor cliques and they continue the process until all the cliques are modified. However, the algorithms do not guarantee to obtain an M$(k+1)$DG and, thus, they require an additional procedure to continue adding cliques of size $k+1$ until an M$(k+1)$DG is constructed. The algorithm proposed in (Proulx and Zhang, 2014), at each step, considers the addition of a vertex to a clique of size k. The vertex must belong to an adjacent clique in the selected junction tree representation and it is selected according to the likelihood. Then, after the addition of the vertex, the junction tree representation is updated by removing any non-maximal clique. This process continues until all the cliques in the junction tree are of size $k+1$. Finally, they transform the obtained decomposable graph into an M$(k+1)$DG by an iterative procedure that adds cliques of size $k+1$ to the separators with a size smaller than k.

1.2. Contributions

In this work, we propose a family of efficient algorithms, called **fractal tree** (**FT**), for learning MkDGs using a **divide-and-conquer** strategy based on the particularities of MkDGs. The algorithms construct a sequence of MiDGs, for $i = 2, ..., k$, in $k-1$ growing steps. At each growing step, the input maximal i-order decomposable graph is decomposed into subgraphs related to its separators. Then the subgraphs are solved using a novel extension of CL, called the generalized Chow-Liu algorithm (see Section 3.2). The solutions to the subgraphs associated to each separator are added to the input structure for generating the next M$(i+1)$DG.

This work proposes two variants of FT (see Section 4): the parallel fractal tree and the sequential fractal tree. **Parallel fractal tree** solves all the the subgraphs associated to the separators in parallel, without considering the interactions among them, while **sequential fractal tree** solves them sequentially, taking into consideration their interactions. Both algorithms have a computational complexity of $\mathcal{O}(k^2 \cdot n^2 \cdot N)$, in the worst case, which is k^2 times the computational complexity of CL implemented using Prim's algorithm with adjacency lists. In the performed experimental comparison, the fractal tree algorithms have shown a competitive performance compared to other state-of-the-art algorithms.

Additionally, we have developed an efficient **prune-and-graft** operator that transforms an MiDG into another MiDG with equal or higher likelihood. It has a computational complexity of $\mathcal{O}(i \cdot n^2 \cdot N)$. This procedure can be used at the end of each growing step of FT in order to improve the obtained structure without increasing its computational complexity (see Section 5).

We consider that the FT family of algorithms is especially advisable to model high dimensional domains due to the following reasons:

- Their computational complexity is quadratic in the number of implied random variables,
- due to their modularity they can be parallelized in terms of separators, and
- they produce a sequence of MiDG for $i = 2, ..., k$ which allow it to select the most appropriate model according to problem-dependent criterion.

The efficient learning of (probabilistic) models for high dimensional domains can be considered a key aspect in massive data analysis (Jordan et al., 2013).

The rest of this work is organized as follows. Section 2 introduces the main theoretical background. We present the decomposable graphs, the decomposable models and the maximum likelihood problem that we face in this work. In Section 3 we present the intuitions and the justification behind the fractal tree algorithms, which include the separator-based decomposition, the separator problem and the generalized Chow and Liu's algorithm. In Section 4 we present the fractal tree family of algorithms. Additionally, two particular implementations of the family of fractal tree algorithms are proposed: parallel fractal tree and sequential fractal tree. Section 5 presents the prune-and-graft procedure. Section 6 summarizes the experimental results obtained by the proposed algorithms in six datasets from the UCI repository and more than 1000 artificial domains. In addition, a procedure for selecting an appropriate value of the parameter k is explained. Finally, in Section 7 we present the conclusion of this work, highlighting the major contributions.

2. Background

We denote by $\boldsymbol{X} = (X_1, ..., X_n)$ an n-dimensional random variable which is distributed according to the probability distribution $p(\boldsymbol{X})$, where X_i is a univariate discrete random variable for $i = 1, ..., n$. We denote an instantiation (a sample) of $\boldsymbol{X}$ by $\boldsymbol{x} = (x_1, ..., x_n)$, where x_i is an instantiation of X_i for $i = 1, ..., n$. Let C be a subset of the indexes $\{1, ..., n\}$ of size $|C|$. The random variable $\boldsymbol{X}_C$ represents the $|C|$-dimensional random variable $(X_i)_{i \in C}$.

A data set $\mathcal{D} = \{\boldsymbol{x}^1, ..., \boldsymbol{x}^N\}$ is a collection of N instances, independent and identically distributed according to $p(\boldsymbol{X})$. Given a data set $\mathcal{D}$, the associated **empirical probability distribution** over $\boldsymbol{X}$ is given by $\hat{p}(\boldsymbol{x}) = N_{\boldsymbol{x}}/N$, where $N_{\boldsymbol{x}}$ is the number of occurrences of $\boldsymbol{x}$ in $\mathcal{D}$. Empirical distributions are given in terms of the observed frequency estimates of the data in a closed form. It should be noted that the empirical distribution is the maximum likelihood distribution given $\mathcal{D}$, i.e., $\hat{p} = \operatorname*{argmax}_q \prod_{i=1}^{N} q(\boldsymbol{x}^i)$.

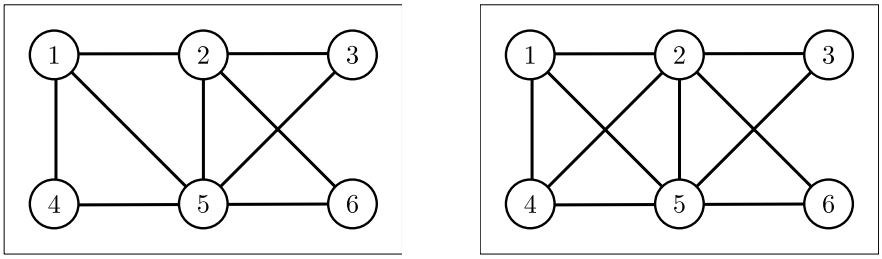
2.1. Decomposable graphs

This section formally defines the decomposable graphs, the candidate edges and the maximal k-order decomposable graphs. We are especially interested in maximal k-order decomposable graphs because they form the structures of the maximum likelihood decomposable models with a bounded clique size.

Let $\boldsymbol{G} = (V, E)$ be an undirected **graph**, where $V = \{1, ..., n\}$ is a set of indexes called the vertices of the graph and E is a set of pairs of vertices $\{u, v\}$ called edges. A graph is said to be complete if it contains every possible edge, i.e., $E = \{\{u, v\} : \{u, v\} \subseteq V\}$. An empty graph is a graph without edges, i.e., $E = \emptyset$. The subgraph induced by $V' \subseteq V$, $\boldsymbol{G}[V'] = (V', E[V'])$, is a graph with the vertex set V', where the set of edges is given by $E[V'] = \{\{u, v\} : \{u, v\} \subseteq V' \wedge \{u, v\} \in E\}$.

Definition 1. *Let $\boldsymbol{G} = (V, E)$ and $\boldsymbol{G}^+ = (V^+, E^+)$ be two undirected graphs. $\boldsymbol{G}^+$ is **coarser** than $\boldsymbol{G}$ (or equivalently, $\boldsymbol{G}$ is **thinner** than $\boldsymbol{G}^+$) if $V = V^+$ and $E \subsetneq E^+$, and it is denoted as $\boldsymbol{G} \prec \boldsymbol{G}^+$.*

For example, $\boldsymbol{G}^+$ is coarser than $\boldsymbol{G}_3$, and $\boldsymbol{G}_3$ is thinner than $\boldsymbol{G}^+$ (see Figure 1).



(a) Maximal 3-order decomposable graph, $\boldsymbol{G}_3$. (b) 4-order decomposable graph, $\boldsymbol{G}^+$.

Figure 1: Examples of decomposable graphs.

Definition 2. *The neighbors of u in $\boldsymbol{G}$ is the set of vertices connected by an edge to u, $\{v \in V : \{u, v\} \in E\}$ and it is denoted by $\mathbb{N}_{\boldsymbol{G}}(u)$. We define the **common neighbors** of a set of vertices S in $\boldsymbol{G}$ as the set of vertices connected by edges to all the vertices in S, $\bigcap_{u \in S} \mathbb{N}_{\boldsymbol{G}}(u)$. The common neighbors of S are denoted by $\mathbb{N}_{\boldsymbol{G}}(S)$ or simply by $\mathbb{N}(S)$, when $\boldsymbol{G}$ it is clear from the context. The subgraph induced by the common neighbors of a set of vertices S is called the **mantle** of S and it is denoted as $\boldsymbol{G}[\mathbb{N}(S)]$.*

For example, $\mathbb{N}_{\boldsymbol{G}_3}(\{2, 5\}) = \{1, 3, 6\}$ and $\mathbb{N}_{\boldsymbol{G}^+}(\{2, 5\}) = \{1, 3, 4, 6\}$ (see Figure 1).

A path is a sequence of distinct vertices connected by edges. A cycle is a path that begins and ends at the same vertex. A chord of a cycle is an edge among two vertices which are not adjacent in the cycle. A connected component of a graph is a subgraph induced by $V' \subseteq V$ in which any two vertices are connected to each other by a path, and are not connected to vertices in $V \setminus V'$ in the original graph. We say that an undirected graph is a **decomposable graph** (**DG**) if for any cycle of length greater than 3 there exists a chord. Decomposable graphs are also known in the literature as chordal graphs or triangulated graphs. In DGs, the tree-width is given by the maximum clique size minus one and the computational complexity for solving inference problems over a model is exponential with respect to its tree-width. A **tree** is a decomposable graph with $n - 1$ edges without cycles. A **forest** is a decomposable graph where each of its connected components is a tree. Note that empty graphs and trees are forests.

Given two non-adjacent indexes u and v, a subset $S \subseteq V \setminus \{u, v\}$ is a separator for u and v, when the graph induced by $V \setminus S$ separates u and v into two different connected components. If no proper subset of S is a separator for u and v, then S is a **minimal separator** for u and v. Any $C \subseteq V$ is called a **maximal clique** for $\boldsymbol{G}$ if the subgraph induced by C, $\boldsymbol{G}[C]$, is a complete graph and there is no proper superset of C which induces a complete subgraph. From here on, we will call the minimal separators and the maximal cliques, separators and cliques, for the sake of brevity.

Let $C_1, ..., C_m$ be a numbered sequence of the set of cliques $\mathbb{C}(\boldsymbol{G})$. Let $S_i = C_i \cap \bigcup_{j=1}^{i-1} C_j$ for $i = 2, ..., m$. The sequence $C_1, ..., C_m$ is said to be a chain of cliques for $\boldsymbol{G}$ if S_i is contained in some clique $C \in \{C_1, ..., C_{i-1}\}$ for any $i = 2, ..., m$. It can be proven that a graph $\boldsymbol{G}$ is a DG if and only if it has a chain of cliques (Lauritzen, 1996). In DGs, the sets S_i for $i = 2, ..., m$ are the (minimal) separators. We denote the set of separators (without repetition) associated to the DG $\boldsymbol{G}$ as $\mathbb{S}(\boldsymbol{G})$.

In this work we propose a constructive procedure that adds edges to a DG while maintaining its decomposability.

Definition 3. *Let $\boldsymbol{G}$ be a decomposable graph. We say that an edge is a **candidate edge** for $\boldsymbol{G}$ if its addition to $\boldsymbol{G}$ produces a decomposable graph.*

The following result, adapted from (Desphande et al., 2001), can be used to characterize the set of candidate edges.

Theorem 4. *Given a decomposable graph $\boldsymbol{G} = (V, E)$, an edge $\{u, v\} \notin E$ is a candidate edge if and only if there exists a minimal separator S for u and v, such that $\{u, v\} \subseteq \mathbb{N}(S)$. The addition of $\{u, v\}$ to $\boldsymbol{G}$ creates the clique $\{u, v\} \cup S$. We denote the set of candidate edges whose addition creates cliques of size k as $\mathbb{E}_k(\boldsymbol{G})$.*

We are interested in two types of decomposable graphs which control explicitly their maximum clique size.

Definition 5. *A k**-order decomposable graph** (k**DG**) is a decomposable graph for which the maximum clique size is k. A **maximal** k**-order decomposable graph** (Mk**DG**) is a kDG for which all the cliques are of size k and the addition of a candidate edge creates a clique of size $k + 1$.*

For example, $\boldsymbol{G}_3$ is an M3DG and $\boldsymbol{G}^+$ is a 4DG (see Figure 1). MkDGs are also known in the literature as $(k-1)$-hypertrees (Srebro, 2000). Note that the empty graph is an M1DG, a forest is a 2DG and a tree is an M2DG. MkDGs are maximal in the sense that no more cliques of size k can be constructed by adding candidate edges. An MkDG has the following interesting structural properties, among others (Malvestuto, 1991):

Theorem 6. *Let $\boldsymbol{G}_k$ be an MkDG. $\boldsymbol{G}_k$ fulfills the following properties*

i) $\mathbb{C}(\boldsymbol{G}_k)$ *is a set of* $m = n - k + 1$ *cliques of size* k*, and*
ii) the size of all the minimal separators in $\mathbb{S}(\boldsymbol{G}_k)$ *is* $k - 1$.

2.2. Decomposable models

This section presents the decomposable models, probabilistic models based on decomposable graphs. Let $\boldsymbol{G}$ be a DG with the set of cliques $\mathbb{C}(\boldsymbol{G})$ and the set of separators $\mathbb{S}(\boldsymbol{G})$. We associate each clique C in the set of cliques $\mathbb{C}(\boldsymbol{G})$ to the $|C|$-dimensional random variable $\boldsymbol{X}_C$ and each separator S in the set of separators $\mathbb{S}(\boldsymbol{G})$ to the $|S|$-dimensional random variable $\boldsymbol{X}_S$. A **decomposable model** (Lauritzen et al., 1984) is given by $\boldsymbol{M} = (\boldsymbol{G}, \mathcal{P}_{\boldsymbol{G}})$, where $\boldsymbol{G}$ is a decomposable graph, and $\mathcal{P}_{\boldsymbol{G}}$ is a set of probabilities associated to the cliques and the separators of $\boldsymbol{G}$, $\mathcal{P}_{\boldsymbol{G}} = \{p(\boldsymbol{X}_C) : C \in \mathbb{C}(\boldsymbol{G})\} \cup \{p(\boldsymbol{X}_S) : S \in \mathbb{S}(\boldsymbol{G})\}$. The probabilities satisfy that they are compatible under marginalization, i.e., $\sum_{x_{C \setminus S}} p(\boldsymbol{x}_C) = \sum_{x_{C' \setminus S}} p(\boldsymbol{x}_{C'}) = p(\boldsymbol{x}_S)$ for any $\{C, C'\} \subset \mathbb{C}(\boldsymbol{G})$ where $C \cap C' = S$. The decomposable model $\boldsymbol{M}$ represents the following factorization of the joint probability distribution $p(\boldsymbol{x})$: $p_{\boldsymbol{M}}(\boldsymbol{x}) = \prod_{C \in \mathbb{C}(\boldsymbol{G})} p(\boldsymbol{x}_C) / \prod_{S \in \mathbb{S}(\boldsymbol{G})} p(\boldsymbol{x}_S)^{d_S}$, were d_S is the number of cliques that contain S minus one (Lauritzen et al., 1984). We call to d_S **degree** of the separator S. We call the decomposable models with kDG and MkDG structures, k-order decomposable models and maximal k-order decomposable models, respectively.

The set of probabilities associated to a maximum likelihood decomposable model is known to be the set of maximum likelihood probabilities (Lauritzen, 1996). Furthermore, the set of maximum likelihood probability distributions can be computed in a closed form for decomposable models (Lauritzen, 1996), avoiding computational intensive iterative approaches such as iterative proportional fitting. The maximum likelihood probability distributions are the empirical

marginal probability distributions. Henceforth, we will concentrate on the structural learning of the decomposable models. With a slight abuse in notation, from here on, we will denote p_M by $p_{\boldsymbol{G}}$ to emphasize that once the use of the empirical probability distributions is decided upon, the likelihood is a function of the graph $\boldsymbol{G}$ only.

A summary of the main theoretical concepts introduced together with its notation can be found in Table 3 (see Appendix A: Theoretical results).

2.3. Learning maximum likelihood k-order decomposable models

This section formally defines the problem of learning maximum likelihood k-order decomposable models and a related problem.

Problem 1. *Let $\mathcal{D} = \{\boldsymbol{x}^1, ..., \boldsymbol{x}^N\}$ be an i.i.d. data set according to an unknown distribution $p(\boldsymbol{X})$. This problem consists of finding a k-order decomposable graph $\boldsymbol{G}$ which maximizes the likelihood of the data:*

$$\boldsymbol{G}^* = \underset{\boldsymbol{G} \in \mathcal{G}_k}{\operatorname{argmax}} \prod_{\boldsymbol{x} \in \mathcal{D}} p_{\boldsymbol{G}}(\boldsymbol{x}) \tag{1}$$

where $\mathcal{G}_k$ denotes the set of k-order decomposable graphs.

Given a decomposable graph $\boldsymbol{G}$ and a coarser structure $\boldsymbol{G}^+$, the likelihood of $\boldsymbol{G}^+$ is equal or higher (Malvestuto, 1991). Therefore, the solution to the problem of learning a maximum likelihood kDG is an MkDG (Malvestuto, 1991). Consequently, our approaches restrict the search of maximum likelihood kDG to the class of MkDGs. Problem 1 is equivalent to finding the MkDG which minimizes the empirical entropy of $\boldsymbol{X}$ distributed according to $p_{\boldsymbol{G}}$, $H_{\boldsymbol{G}} = \frac{1}{N} \prod_{\boldsymbol{x} \in \mathcal{D}} p_{\boldsymbol{G}}(\boldsymbol{x})$ (Malvestuto, 1991). From here on we will deal with Problem 1 through the minimum entropy formulation.

The learning of a maximum likelihood MkDG for $k = 2$ can be solved polynomially using CL (Chow and Liu, 1968). This is significantly different than the general case of $k > 2$ where the problem is NP-hard (Srebro, 2000, 2003). In order to deal with Problem 1, we propose tractable suboptimal algorithms with a computational complexity of $\mathcal{O}(k^2 \cdot n^2 \cdot N)$, in the worst case.

Next, we define a problem related to Problem 1.

Problem 2. *Given an MkDG, $\boldsymbol{G}_k$, and a data set $\mathcal{D}$, find a maximum likelihood M$(k+1)$DG coarser than $\boldsymbol{G}_k$.*

If an algorithm can solve Problem 2, its recursive application would produce a sequence of MiDGs for $i = 2, ..., k$, $\boldsymbol{G}_2 \prec \boldsymbol{G}_3 \prec ... \prec \boldsymbol{G}_k$ (see Corollary 11 and the comments below), where $\boldsymbol{G}_k$ could be a good approximation to Problem 1. Unfortunately, Problem 2 still remains NP-hard for $k > 1$ (see Corollary 12).

3. A divide-and-conquer strategy

This section formally presents the intuitions behind FT. The theoretical justification can be found in the Appendix.

FT follows a constructive procedure based on a two-fold divide-and-conquer strategy. First, FT tries to construct a sequence of MiDGs for $i = 2, ..., k$, each one coarser than the previous one. In other words, it decomposes Problem 1 into a sequence of $k - 1$ growing steps which correspond to instances of Problem 2, where the solution to the i^{th} instance of the problem is

the starting structure of the next. As we noted in the previous section, Problem 2 is still NP-hard. However, our goal is to propose an efficient approach to Problem 1 and FT is based on an approximation to Problem 2 with a computational complexity of $\mathcal{O}(i \cdot n^2 \cdot N)$ for an input MiDG.

It should be noted that, in order to obtain a competitive approach to Problem 1, we do not have to obtain the optimal solution to Problem 2 at each growing step. Given an MiDG thinner than the optimal MkDG to Problem 1, we need to find an M$(i+1)$DG thinner than the MkDG. In other words, at step i, we need to add candidate edges to the input MiDG which are included in the optimal MkDG. In this sense, in order to obtain the target MkDG, different sequences of MiDGs for $i = 2, .., k$ can be constructed. FT tries to efficiently find one of these sequences.

3.1. Separator-based decomposition

In this section we introduce the separator-based decomposition, which can be used to deal with Problem 2. This decomposition is the basis for the growing steps of FT. At the i^{th} growing step, the separator-based decomposition is used in order to i) efficiently identify the candidate edges that create cliques of size $(i+1)$ and ii) decompose Problem 2 into a set of subproblems related to the separators which can be efficiently solved.

Given an input MiDG, $\boldsymbol{G}_i$, in order to obtain a coarser M$(i+1)$DG, we need to construct a sequence of coarser $(i+1)$DGs. Thus, in order to deal with Problem 2, we need to identify all the candidate edges that create cliques of size i for $\boldsymbol{G}_i$ and for any $(i+1)$DG coarser than $\boldsymbol{G}_i$.

As we indicated in Theorem 4, the identification of the set of candidate edges can be efficiently performed using the separators, which motivates the next definition.

Definition 7. *Let S be a separator of a DG $\boldsymbol{G}$. The set of candidate edges due to S in $\boldsymbol{G}$ is defined as*

$$\begin{aligned}\mathbb{E}_{\boldsymbol{G}}(S) &= \mathbb{E}_2(\boldsymbol{G}[\mathbb{N}(S)]) \\ &= \{\{u,v\} : u \text{ and } v \text{ are at distinct connected components in } \boldsymbol{G}[\mathbb{N}(S)]\}\end{aligned}$$

The set of candidate edges due to a separator is composed by the candidate edges that create cliques of size 2 in its mantle. This set can be efficiently determined by the set of pairs of vertices which belong to different connected components of the mantle.

We call **separator-based decomposition** for a graph $\boldsymbol{G}$ induced by a set of separators $\mathcal{S}$ to the set of mantles due to $\mathcal{S}$, $\{\boldsymbol{G}[\mathbb{N}(S)] : S \in \mathcal{S}\}$. This decomposition can be used to efficiently determine the set of candidate edges for a decomposable graph due to a set of separators. Note that by selecting $\mathcal{S} = \mathbb{S}(\boldsymbol{G})$ we can determine the set of candidate edges for $\boldsymbol{G}$. In order to efficiently determine the set of candidate edges due to a separator, we suggest characterizing the mantle by the sets of vertices associated to their connected components.

By Theorem 4, we know that the addition of a candidate edge due to S creates a clique of size $|S|+2$. Thus, the size of the separator can be used to effectively control the size of the created cliques. In order to deal with Problem 2, we need to identify the candidate edges that create cliques of size $i+1$ and, therefore, we need to identify the set of separators of size $i-1$ for $\boldsymbol{G}_i$ and for any particular $(i+1)$DG coarser than $\boldsymbol{G}_i$, $\boldsymbol{G}^+$. For $\boldsymbol{G}_i$ this subset is its set of separators, $\mathbb{S}(\boldsymbol{G}_i)$ (see Theorem 6). Furthermore, by Corollary 16, we know that the set of separators of size $i-1$ for $\boldsymbol{G}^+$ is contained in $\mathbb{S}(\boldsymbol{G}_i)$. Consequently, in order to determine the set of candidate edges for $\boldsymbol{G}_i$ (and $\boldsymbol{G}^+$) that create cliques of size $i+1$, it is enough to perform the separator-based decomposition for $\boldsymbol{G}_i$ (or $\boldsymbol{G}^+$) induced by $\mathbb{S}(\boldsymbol{G}_k)$ (see Proposition 17).

It should be noted that the separator-based decomposition induced by $\mathbb{S}(\boldsymbol{G}_i)$ decomposes $\boldsymbol{G}_i$ and $\boldsymbol{G}^+$ into mantles with a forest structure. Moreover, all the mantles for $\boldsymbol{G}_i$ are empty graphs while the mantles for $\boldsymbol{G}_i$ are forest in general (see Proposition 18), which includes empty graphs and trees. Using this decomposition, Problem 2 can be approximated by dividing it into a set of separator problems.

Problem 3. ***[The separator problem]*** *Let $\boldsymbol{G}$ be an MiDG or a coarser $(i+1)$DG and let $\mathcal{D}$ be a data set. Let S be a separator of $\boldsymbol{G}$ of size $i-1$ where its mantle $\boldsymbol{G}[\mathbb{N}(S)]$ is a forest. Find the tree coarser than the mantle of S that produces the subgraph induced by $\mathbb{N}(S) \cup S$, $\boldsymbol{G}[\mathbb{N}(S) \cup S]$, which maximizes the likelihood.*

It should be highlighted that the mantle of every separator of size $i-1$ in $\boldsymbol{G}$ is a forest (see Proposition 18). From here on, for the sake of brevity, we say that a separator is solved when its associated separator problem is solved.

In summary, given an input MiDG $\boldsymbol{G}_i$, FT decomposes Problem 2 into a set of separator problems using the separator-based decomposition induced by $\mathbb{S}(\boldsymbol{G}_i)$. Then, FT solves every separator by creating a tree in its mantle. Finally, it joins the partial solutions to each separator problem to form the output M$(i+1)$DG used to approximate Problem 2[2].

3.2. The generalized Chow-Liu algorithm

In this section we present a natural extension of the Chow-Liu algorithm (CL, Chow and Liu (1968)) called **the generalized Chow-Liu algorithm** (**GCL**), which solves the separator problem. GCL creates a maximum weighted tree in the mantle of the separator S using a maximum spanning tree solver. The weight assigned to a candidate edge $\{u, v\}$ due to S is given by the empirical conditional mutual information $\hat{I}(X_u, X_v | \boldsymbol{X}_S)$ (see Proposition 10). This procedure is equivalent to CL with the appropriate weights and it solves Problem 3. For a review of other extensions of CL we refer the reader to (Højsgaard et al., 2012).

Given the required mutual information quantities, a simple and efficient maximum spanning tree solver for dense graphs[3] is Prim's algorithm with adjacency lists, which has a computational complexity of $\mathcal{O}(n^2)$. Prim's algorithm constructs a tree sequentially by adding the edge with the highest weight from a vertex in the tree to a vertex not connected to the tree. This algorithm can be easily adapted to deal with a forest instead of an empty graph by considering the vertices in one of the connected components of the input forest as a unique vertex. In (Eisner, 1997) the reader can find other alternatives to Prim's algorithm with adjacency lists, such as Kruskal's algorithm or Prim's algorithm with Fibonacci heaps, that can also be used in order to implement GCL. We would like to point out that, for dense graphs, Prim's algorithm with Fibonacci heaps has the same complutational complexity as Prim's algorithm with adjacency lists. However, since it is harder to implement, we recommend the use of Prim's algorithm with adjacency lists.

Due to the low computational complexity of the selected maximum spanning tree solver, the computational complexity of GCL is dominated by the computation of the required $\mathcal{O}(n^2)$ conditional mutual information quantities. A straightforward computation of the empirical conditional mutual information $\hat{I}(X_u, X_v | \boldsymbol{X}_S)$ with $|S| = i-1$ has a complexity of $\mathcal{O}(i \cdot N + 2^i)$ (for binary random variables), which is exponential in i. However, assuming that $2^i < N$, we have a complexity of $\mathcal{O}(i \cdot N)$. We would like to emphasize that the assumption of $2^i < N$ is

[2]We would highlight that the union of the partial solutions added to the MiDG usually does not form a tree.

[3]Graphs that consider $\mathcal{O}(n^2)$ edges for constructing a maximum spanning tree.

necessary if we do not want to suffer from the overfitting phenomenon (see Section 6). Since GCL requires to compute $\mathcal{O}(n^2)$ empirical conditional mutual informations, its computational complexity is $\mathcal{O}(i \cdot n^2 \cdot N)$. In order to speed up the implementation of GCL, we recommend parallelizing the computation of the conditional mutual informations following a procedure such as that proposed in (Madsen et al., 2014).

3.3. Generation of new candidate edges

The addition of a candidate edge to a decomposable graph can cause the generation of new candidate edges. Thus, the order in which candidate edges are added can be crucial in order to solve an instance of Problem 2.

Definition 8. *Let $\boldsymbol{G}$ be an MiDG or a coarser $(i+1)$DG, let S and S' be two separators of $\boldsymbol{G}$ with size $i+1$ where $\{u\} = S' \setminus S$. We say that S' is **adjacent** to S if $u \in \mathbb{N}_{\boldsymbol{G}}(S)$.*

It should be noted that the relation is symmetric[4] and, thus, we say that S and S' are adjacent. We say that a separator S **blocks** two separators S' and S'', if every sequence of adjacent separators starting in S' and ending S'' contains S.

In terms of separators, the addition of a candidate edge due to a separator can only modify the mantles of **adjacent separators** (see Proposition 19). The addition can cause the set of candidate edges due to the adjacent separator to increase (never decrease): it can donate a vertex from its mantle by connecting it to a vertex of the mantle of the adjacent separator (see Proposition 19 for further details). In other words, the addition of an edge to the mantle of a separator only can change the problem associated to an adjacent separator. Therefore, the ordering in which the separator problems are solved can have a great impact in the obtained solution to Problem 2.

4. Fractal tree algorithms

Next, we propose two particular FT algorithms: parallel fractal tree and sequential fractal tree[5]. On the one hand, parallel fractal tree represents the most efficient algorithm and, on the other hand, sequential fractal tree obtains better results for dealing with Problem 1.

Algorithm 1. *(Parallel fractal tree)*
Input: *A data set $\mathcal{D} = \{\boldsymbol{x}^1, ..., \boldsymbol{x}^N\}$, a positive integer $k \leq n$.*
Output: *An MkDG.*
Pseudocode:

1. *Initialize the empty graph* $\boldsymbol{G}_1$.
2. *For* $i = 1, ..., k-1$:
3. -*For each separator* $S \in \mathbb{S}(\boldsymbol{G}_i)$, *do (**parallel growing step**):*
4. *Apply GCL algorithm to the mantle of* S *for obtaining* E^S.
5. -*Add* E^S *to* $\boldsymbol{G}_{i+1}$ *for every* $S \in \mathbb{S}(\boldsymbol{G}_i)$.
6. -$\boldsymbol{G}_{i+1} := \boldsymbol{G}_i$.
7. -*Obtain the separators of* $\boldsymbol{G}_{i+1}$.
8. *Return* $\boldsymbol{G}_k$.

[4] i.e., if S' is adjacent to S then S is adjacent to S'

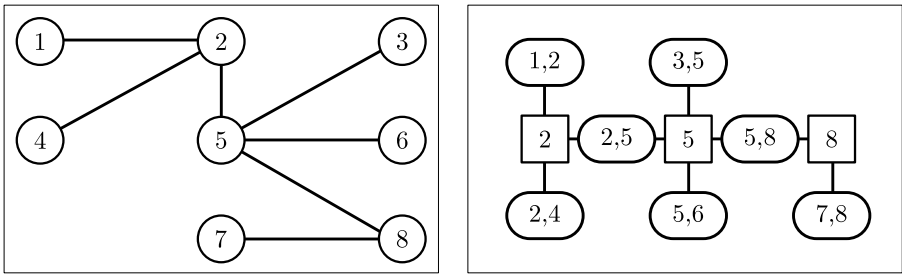
[5] The implementation in Python of the proposed algorithms is publicly available at https://bitbucket.org/AritzPerez/fractaltree.

4.1. Parallel fractal tree

The pseudo-code of **parallel fractal tree** (**PFT**) is given in Algorithm 1. At each parallel growing step (lines 3-5), this algorithm solves the separator problems ignoring the interactions among the mantles of different separators (see Proposition 19). It should be noted that, at the i^{th} growing step, given an MiDG, $\boldsymbol{G}_i$, PFT achieves one of the maximum likelihood M$(i+1)$DGs that can be constructed by adding a subset of the set of candidate edges $\mathbb{E}_{i+1}(\boldsymbol{G}_i)$ to $\boldsymbol{G}_i$.

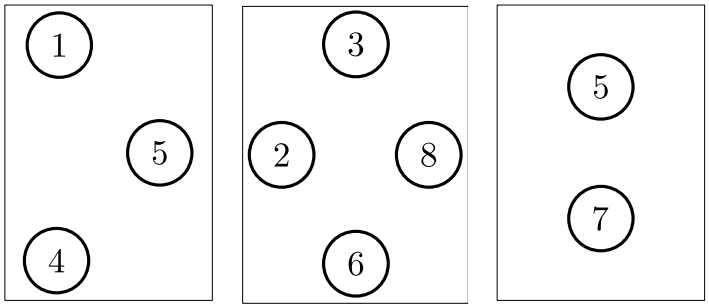
In PFT, the growing step (Algorithm 1, lines 3-4) can be performed in parallel for each separator because the set of candidate edges considered for each separator S is static. That is, its set of candidate edges $\mathbb{E}(S)$ is not updated by the addition of edges to the mantles of other separators. It should be noted that, for mantles with only two vertices, the computation of the empirical conditional mutual information can be avoided because it has a single candidate edge.

4.2. An example with PFT



(a) Undirected graph representation of $\boldsymbol{G}_2$. (b) Edge-separator graph representation of $\boldsymbol{G}_2$.

Figure 2: This figure shows the M2DG, $\boldsymbol{G}_2$, obtained after the first growing step of both PFT and SFT. Two alternative representations are shown: undirected graph and edge-separator graph representation (Malvestuto, 2012). In the edge-separator graph representation, we have represented the cliques by circles and the separators by squares.

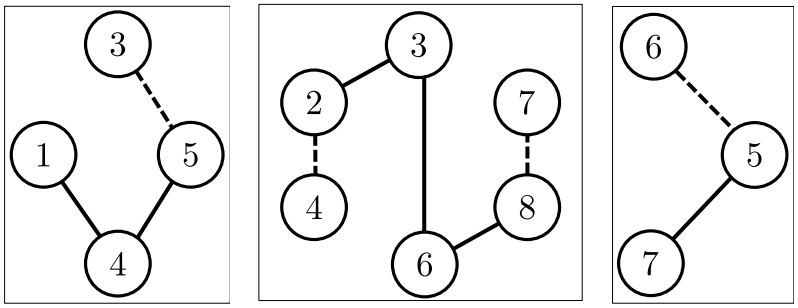


(a) The mantle of $\{2\}$. (b) The mantle of $\{5\}$. (c) The mantle of $\{8\}$.

Figure 3: This figure shows the mantles of the separators of the M2DG, $\boldsymbol{G}_2$, shown in Figure 2. Note that all the mantles are empty subgraphs.

Figures 2, 3, 4 and 5 illustrate an execution in a domain with $n = 8$ random variables for a maximum clique size of $k = 3$. The first parallel growing step is equivalent to CL because the empty graph is the M1DG, which has the empty set as the unique separator. The second parallel growing step divides the obtained M2DG, shown in Figure 2, into the mantles associated to the

separators $\mathbb{S}(\boldsymbol{G}_2) = \{\{2\}, \{5\}, \{8\}\}$. The corresponding mantles are shown in Figures 3a, 3b and 3c, respectively. Once the mantles are determined, GCL is applied to solve each separator problem without taking into account the interactions among them. The obtained results are illustrated in Figures 4a, 4b and 4c, respectively. Finally, the solutions to each separator problem are gathered together to form the attained M3DG, shown in Figure 5.

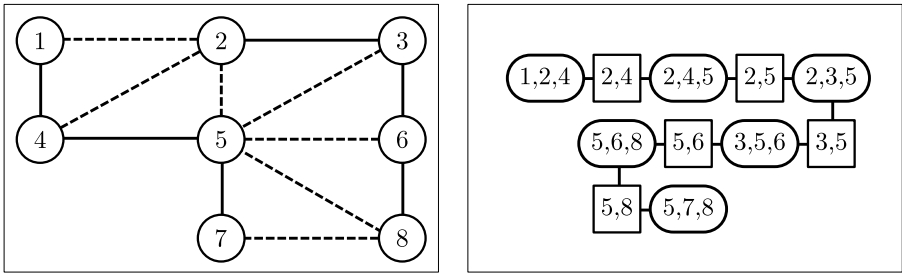


(a) The solution to the mantle of $\{2\}$.

(b) The solution to the mantle of $\{5\}$.

(c) The solution to the mantle of $\{8\}$.

Figure 4: This figure shows the solutions, given by the PFT algorithm, to the separator problems represented in Figure 3. Note that all the solutions correspond to trees. The solid lines represent the edges added during the parallel growing step to each mantle by GCL, while the dashed lines represent the edges that appear in the mantles due to the addition of edges in the mantles of adjacent separators.



(a) Undirected representation of $\boldsymbol{G}_3^{PFT}$.

(b) Edge-separator graph representation of $\boldsymbol{G}_3^{PFT}$.

Figure 5: This figure shows the solution obtained by PFT, $\boldsymbol{G}_3^{PFT}$, which has been created by adding the partial solutions to the mantles of each separator of $\boldsymbol{G}_2$ (see Figure 4). The solid lines in Figure 5a represent the edges added in the last growing step while the dashed lines represent the edges added in the previous step.

PFT adds a forest at each growing step (see Figure 5a). In consequence, on average, as i increases the degree of the separators tends to decrease in the sequence of MiDGs constructed by PFT. Thus, as i increases, PFT has a natural tendency towards separators of lower degree. We call this phenomenon the **chain caveat**. This tendency could influence the behavior of the algorithm for approximating some instances of Problem 1[6]. The effect of the chain caveat can be intensified if the empirical mutual information quantities have low information (see Section 6.2). The chain caveat is a direct consequence of the expected degree of the vertex of a tree, which is 2.

[6]e.g., instances whose optimal solutions have separators of high degree

Furthermore, the addition of a forest at each growing step limits the structures that can be learned by PFT: it can not attain decomposable graphs with cliques containing more than two separators. The generated cliques and separators are related to the edges and separators of the added forest, respectively. Since each edge of the forest connects two vertices and the separators are a subset of the vertices, each clique can not contain more than two separators. An example of unattainable structure is shown in Figure 7, where the clique $\{2, 4, 5\}$ contains the separators $\{2, 4\}$, $\{2, 5\}$, and $\{4, 5\}$. We call this phenomenon the **unattainable structure caveat**. In the performed experimentation with artificial domains, we have randomly sampled MkDGs using Algorithm 5, and almost all of the generated structures are unattainable. However, PFT has shown a competitive behavior even in these domains.

4.3. Sequential fractal tree algorithm

The **sequential fractal tree** (**SFT**) algorithm increases the number of candidate edges considered at each growing step with respect to PFT and avoids the chain and the unattainable structure caveats. Assuming that the i^{th} step of PFT and SFT starts from the same MiDG structure, the growing step of SFT obtains an M$(i+1)$DG structure with a likelihood equal to or higher than that obtained by PFT. The pseudo-code of SFT is shown in Algorithm 2.

In SFT, the separators are solved sequentially using GCL (Algorithm 2, lines 4-6). After the application of GCL to a separator, the mantles of the adjacent separators are modified (Proposition 19) and, therefore, the set of candidate edges of the adjacent separators can increase. The increase of the size of the mantles can produce an exponential increase in the possible solutions (trees) that can be generated in the mantle. In addition, while the parallel growing step adds forests to the input MkDG, the sequential growing can add more general decomposable structures, which avoids the unattainable structure caveat. For example, Figure 7 shows a structure where the clique $\{2, 4, 5\}$ contains three separators, $\{2, 4\}$, $\{2, 5\}$ and $\{4, 5\}$.

Algorithm 2. *(Sequential fractal tree)*
Input: *A data set $\mathcal{D} = \{\boldsymbol{x}^1, ..., \boldsymbol{x}^N\}$, a positive integer k.*
Output: *An MkDG.*
Pseudocode:

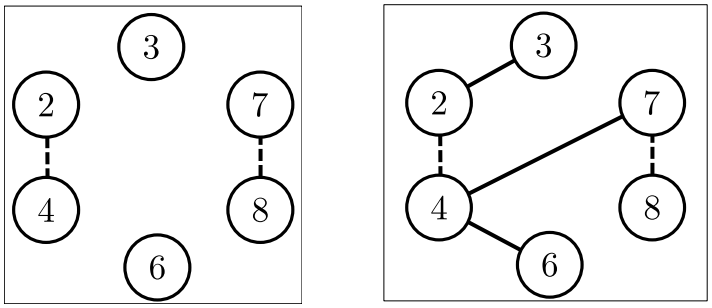
1. *Construct* $\mathbb{S}(\boldsymbol{G}_1)$ *for the empty graph* $\boldsymbol{G}_1$*.*
2. *For* $i = 1, ..., k-1$*:*
3. *-Sort the separators*[7] *in* $\mathbb{S}(\boldsymbol{G}_i)$ *for* $\boldsymbol{G}_i$*.*
4. *-*$\boldsymbol{G}_{i+1} := \boldsymbol{G}_i$*.*
5. *-For* $S \in \mathbb{S}(\boldsymbol{G}_i)$ *in order, do (**sequential growing step**):*
6. *Apply GCL to the mantle of* S *in* $\boldsymbol{G}_{i+1}$ *for obtaining* E*.*
7. *Add* E *to* $\boldsymbol{G}_{i+1}$ *(Proposition 19).*
8. *-Obtain the separators of* $\boldsymbol{G}_{i+1}$*.*
9. *Return* $\boldsymbol{G}_k$*.*

The sequential growing step requires defining an order among the separators for the application of the GCL algorithm. Different criteria can be used to sort the separators. In the performed experimentation, we have observed that sorting the separators by the number of vertices in their mantles in ascending order tends to consider more candidate edges. It should be noted that the solution to a separator increases the size of the mantle of its adjacent separators. In addition, the

[7] e.g., by the number of connected components of their associated subgraph.

number of candidate edges of a separator grows quadratically with the number of vertices in its mantle and the possible trees that can be learned grows exponentially with this number. Consequently, roughly speaking, by solving the separators with smaller mantles first we are increasing the size of the bigger mantles. Therefore, using this criterion, SFT tends to consider more arcs and it can attain more structures due to their quadratic and exponential growth, respectively. In the experimentation we have observed that the criterion based on the size of the mantle has obtained the best results for approximating Problem 1, among the sorting criterion considered.

4.4. An example with SFT



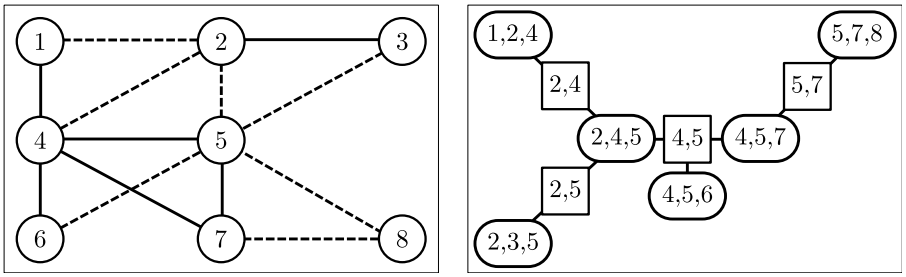
(a) The mantle of $\{5\}$. (b) The solution the mantle of $\{5\}$.

Figure 6: This figure shows the mantle of $\{5\}$ due to the interactions with the mantles of the separators $\{2\}$ and $\{8\}$. The dashed lines represent the edges added due to the interaction with other separators.

Figures 2, 3, 4, 6 and 7 illustrate an execution of SFT. The first ($i = 1$) sequential growing step (lines 4-6, Algorithm 2) is equivalent to PFT and the obtained structure is shown in Figure 2. The second ($i = 2$) sequential growing step divides the M2DG obtained in the previous step into the mantles associated to the separators $\{2\}$, $\{5\}$ and $\{8\}$. The obtained mantles are shown in Figures 3a, 3b and 3c. At this point, the behavior of SFT starts to be different to PFT. First, the order in which the separator problems are solved is decided. In this work, we sort the separators according to the number of connected components of their mantles in ascending order. The number of components of the mantles of $\{2\}$, $\{5\}$ and $\{8\}$ are 3, 4 and 2, respectively. First, SFT solves the separator $\{8\}$, which has the same solution as in PFT (see Figure 4c). However, the addition of the edge $\{5, 7\}$ interacts with the mantle of separator $\{5\}$, as $\{5\}$ is contained in the new clique $\{8\} \cup \{5, 7\}$ (see Proposition 19). Then the separator $\{2\}$ is solved. Since the addition of the edges to the mantle of $\{8\}$ does not produce changes in the mantle of $\{2\}$, the same solution as PFT is obtained for $\{2\}$ (see Figure 4a). In this case, the addition of the edge $\{4, 5\}$ modifies the mantle of separator $\{5\}$, as $\{5\}$ is contained in $\{8\} \cup \{4, 5\}$. Finally, the mantle of the separator $\{5\}$ shown in Figure 6a is solved. The solution is shown in Figure 6b. The edge $\{4, 7\}$ is added instead of $\{6, 8\}$, which is added by PFT. At the end of the execution, SFT obtains the structure shown in Figure 7.

4.5. Computational complexity

In the worst case, PFT and SFT have the same computational complexity. As noted at the end of Section 3.2, the computational complexity of the GCL algorithm is dominated by the computation of the required mutual information quantities. As a direct consequence, the computational



(a) Undirected representation of $\boldsymbol{G}_3^{SFT}$. (b) Edge-separator graph representation of $\boldsymbol{G}_3^{SFT}$.

Figure 7: This figure shows the solution obtained by SFT, $\boldsymbol{G}_3^{SFT}$, which has been created by adding the partial solutions to the mantles of each separator of $\boldsymbol{G}_2$ (see Figures 3a, 6a and 3c). The solid lines represent the edges added in the last growing step while the dashed lines represent the edges added in the previous step.

complexity of the growing step i is also dominated by the computation of the required mutual information quantities. By Corollary 21, the worst case corresponds to an MiDG with a single separator. For this case, the parallel and sequential growing steps require computing $\binom{n-i+1}{2}$ mutual information quantities. Therefore, the overall computational complexity of PFT and SFT is $\mathcal{O}(k^2 \cdot n^2 \cdot N)$, in the worst case. It should be noted that the worst case corresponds to a sequence of MiDGs $\boldsymbol{G}_i$ for $i = 1, ..., k-1$, where $\mathbb{S}(\boldsymbol{G}_i)$ is composed by a single separator, which is overwhelmingly improbable.

In addition, the parallel growing step can be completely parallelized solving each separator problem independently because PFT does not take into account the interactions among the mantles. Furthermore, as we noted in Section 3.2, GCL can be also parallelized, for instance, following (Madsen et al., 2014).

The sequential growing step of SFT can be also parallelized, however, there is a trade-off between the degree of parallelization and the number of interactions among the separators considered. In order to achieve a good trade-off, we recommend creating sets of non-adjacent separators (see Definition 8 for the notion of adjacent separators). The number of sets of separators, s, required is given by the maximum number of separators adjacent to a given one, in the worst case. In the case of MkDGs, it is always possible to create k sets of non-adjacent separators. For instance, for the M2DG shown in Figure 2 we can construct $\mathcal{S}_1 = \{\{2\}, \{8\}\}$ and $\mathcal{S}_S = \{\{5\}\}$ ($s = 2 = k$), for the M3DG shown in Figure 1a we can construct $\mathcal{S}_1 = \{\{2,4\}, \{3,5\}, \{5,8\}\}$ and $\mathcal{S}_2 = \{\{2,5\}, \{5,6\}\}$ ($s = 2 < k$) and for the M3DG shown in Figure 7 we can construct $\mathcal{S}_1 = \{\{2,4\}, \{5,7\}\}$, $\mathcal{S}_2 = \{\{2,5\}\}$ and $\mathcal{S}_3 = \{\{4,5\}\}$ ($s = 3 = k$). Therefore, given an MkDG a sequential growing step can be parallelized in k steps, in the worst case, which is independent from its number of vertices n. That is, first we can solve in parallel all the separators of $\mathcal{S}_1$, then we can solve $\mathcal{S}_2$ in parallel and we can continue this iterative process until we solve in parallel all the separators in $\mathcal{S}_s$. Once, the sets of separators are constructed, we can decide the order in which they are solved. We recommend using a criteria based on the average square degree of the separators contained in each set of separators, where the sets are solved from the lowest values to the highest.

5. A prune-and-graft operator for MkDGs

In this section we present a **prune-and-graft** operator (**P&G**) for MkDG structures, by extending the concept of a leaf vertex from trees to MkDGs. This procedure is used in order to improve the likelihood of the structures obtained by the fractal tree algorithms at the end of each growing step. P&G exploits the structural constraints of MkDGs. The procedure consists of removing each leaf vertex from its separator (prune) to insert into the separator (graft) that maximizes the likelihood. In summary, P&G transforms the input MkDG into another MkDG with an equal or higher likelihood by moving the leaf vertices. Next, we define the terms leaf (vertex) and stem (separator) in order to simplify the description of the P&G procedure.

Definition 9. *Let $\boldsymbol{G}_k$ be an MkDG. A vertex u is called a **leaf** for a $\boldsymbol{G}_k$, when it is not included in any of the separators of $\boldsymbol{G}_k$, $u \in V \setminus \bigcup \mathbb{S}(\boldsymbol{G}_k)$. A separator is called **stem** for $\boldsymbol{G}_k$, when all but one of the vertices in its mantle are leaf vertices, $|\mathbb{N}(S) \setminus \bigcup \mathbb{S}(\boldsymbol{G}_k)| = |\mathbb{N}(S)| - 1$.*

In other words, a leaf of an MkDG belongs to a single clique C. A leaf can be removed from an MkDG without creating additional separators. Thus, when a leaf is removed from an MkDG we obtain a smaller MkDG. Then, if we create a new clique by adding the pruned leaf to a separator of the obtained MkDG, we create another MkDG of the same size as the original MkDG (see Proposition 24). A stem is a separator with a single non-leaf vertex. It is used to designate a separator that can lose the condition of being separator due to the effect of the P&G procedure and, thus, it can be removed from the set of separators.

Algorithm 3. *(Prune-and-graft procedure)*
Input: *The MkDG $\boldsymbol{G}_k = (V, E)$ and a data set $\mathcal{D}$.*
Output: *An MkDG.*
Pseudocode:

1. $\mathcal{L} := \emptyset$, $treated := \emptyset$
2. For $S \in \mathbb{S}(\boldsymbol{G}_k)$ (**find the stems**)
3. -If $|\mathbb{N}(S) \setminus \bigcup \mathbb{S}(\boldsymbol{G}_k)| = |\mathbb{N}(S)| - 1$
4. $\mathcal{L} := \mathcal{L} \cup \{S\}$
5. Sort $S \in \mathcal{L}$ in ascending order of $|\mathbb{N}(S)|$
6. For $S \in \mathcal{L}$ (**P&G the stems**)
7. -For $u \in (\mathbb{N}(S) \setminus \bigcup \mathbb{S}(\boldsymbol{G}_k)) \setminus treated$
8. $treated := treated \cup \{u\}$
9. $S^* := \underset{S' \in \mathbb{S}(\boldsymbol{G}_k)}{\operatorname{argmax}} I(X_u; \boldsymbol{X}_{S'})$
10. $E := (E \setminus \{\{u, v\} : v \in S\}) \cup \{\{u, w\} : w \in S^*\}$
11. -If $|\mathbb{N}(S)| = 1$, then (**remove a stem**)
12. Remove S from $\mathbb{S}(G_k)$
13. For $S' \in \mathcal{S}$ (**find a new stem**)
14. -If $|\mathbb{N}(S') \setminus \bigcup \mathbb{S}(\boldsymbol{G}_k)| = |\mathbb{N}(S')| - 1$
15. Append S' to $\mathcal{L}$
16. $\mathcal{S} := \mathcal{S} \setminus \{S'\}$; Go to (6)
17. $\mathcal{S} := \mathbb{S}(\boldsymbol{G}_k) \setminus \mathcal{L}$
18. For $S \in \mathcal{S}$ (**P&G the rest of separators**)
19. -For $u \in (\mathbb{N}(S) \setminus \bigcup \mathbb{S}(\boldsymbol{G}_k)) \setminus treated$
20. $S^* := \underset{S' \in \mathbb{S}(\boldsymbol{G}_k)}{\operatorname{argmax}} I(X_u; \boldsymbol{X}_{S'})$
21. $E := (E \setminus \{\{u, v\} : v \in S\}) \cup \{\{u, w\} : w \in S^*\}$
22. return the MkDG (V, E)

The pseudo-code of P&G is given in Algorithm 3. The P&G procedure is divided in two parts. In the first part (lines 1-17), all the stems are pruned and grafted. This part starts by identifying the stems (lines 2-4). The stems are treated in a different way because, as we noted before, they can be removed by the P&G procedure. If all the leaves belonging to a stem are pruned and grafted into other separators, the stem only has a single vertex in its mantle. Therefore, it loses its condition of being a separator in the structure and it is removed (lines 11-12). The removal of a stem can cause the creation of a new one (see lines 13-16). In the second part (lines 17-22), the rest of the separators are considered. In this case, the mantles of the separators have at least two vertices that are not leaves. Thus, the separators can not be removed because the size of their mantles can not be smaller than 2.

The pruning of the leaf u from the separator S, reduces the entropy of the structure in $\hat{H}(X_u) - \hat{I}(X_u; \boldsymbol{X}_S)$, where $\hat{H}$ and $\hat{I}$ denote the empirical entropy and empirical mutual information, respectively. Grafting u into S' increases the entropy of the structure in $\hat{H}(X_u) - \hat{I}(X_u; \boldsymbol{X}_{S'})$. Thus, the optimal pruning and grafting of leaf u (line 9,20) consists of pruning from its separator S and grafting into the separator $S^* = \underset{S' \in \mathbb{S}(\boldsymbol{G}_k)}{\operatorname{argmax}} \hat{I}(X_u; \boldsymbol{X}_{S'})$ (see Proposition 24). Therefore, P&G transforms an MkDG into another MkDG with an equal or higher likelihood. Note that, when $S = S^*$, the prune-and-graft process has no effect over the MkDG. At the end of the first step, the P&G operator is not applied because the obtained M2DG is optimal from the likelihood point of view (line 8, Algorithm 2).

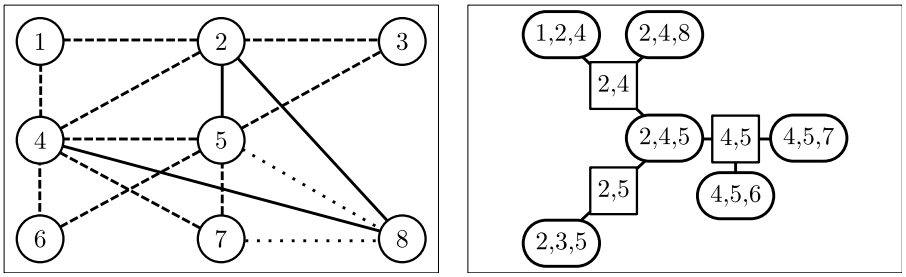
In the worst case, there are $\mathcal{O}(n)$ leaf vertices and $\mathcal{O}(n)$ separators, where each leaf vertex is considered once. Thus, $\mathcal{O}(n^2)$ empirical mutual informations are computed which, assuming that $k \cdot N > 2^k$, has a computational complexity of $\mathcal{O}(k \cdot n^2 \cdot N)$. The leaf separators are sorted by the size of the mantle, which requires $\mathcal{O}(n \log n)$ computations. Therefore, the computational complexity in the worst case of the P&G procedure is $\mathcal{O}(k \cdot n^2 \cdot N)$. It should be noted that the M2DG obtained at the end of the first growing step of the fractal tree algorithms is optimal and the P&G procedure has no effect in that structure and, thus, the P&G procedure is applied $k-2$ times in fractal tree algorithms. Therefore,, the computational complexity of the application of the P&G procedure at the end of the last $k-2$ growing steps (MiDGs for $i = 3, ..., k$) is $\mathcal{O}(k^2 \cdot n^2 \cdot N)$. In consequence, in the worst case, PFT and SFT with the P&G procedure remains with a computational complexity of $\mathcal{O}(k^2 \cdot n^2 \cdot N)$.

5.1. An example with P&G

Figures 7 and 8 illustrate the effect of the P&G procedure (Algorithm 3). P&G takes the M3DG $\boldsymbol{G}_3^{SFT}$ shown in Figure 7 as input and identifies the stems $\{2, 4\}$, $\{2, 5\}$ and $\{4, 5\}$. Then it considers the pruning and grafting of their leaves $\{1, 3, 6, 8\}$. In this case, P&G only prunes the leaf vertex 8 from the common neighbors of $\{5, 7\}$ and grafts it in the separator $\{2, 4\}$. Since the common neighbors of $\{5, 7\}$ has 4 as its only vertex, the separator is removed. Then, the separator $\{4, 5\}$ becomes a new stem and vertex 7 a new leaf (see Figure 8).

6. Experimentation

This section presents a set of experiments which provide evidence about the effectiveness of the proposed algorithms for approximating Problem 1. The results obtained in the experimentation are summarized using the following measures:



(a) Undirected graph representation (b) Edge-separator graph representation

Figure 8: This figure shows the structure obtained after the prune-and-graft operator is applied to $\boldsymbol{G}_3^{SFT}$ (see Figure 7). The solid lines represent the edges added by the P&G procedure, the dotted lines the removed edges and the dashed lines the edges from $\boldsymbol{G}_3^{SFT}$ that are preserved.

- **Power of fitting**: Measures the capability of the learned models to fit (explain or comprise) the available data. It is the likelihood of the training set (given the model), divided by the product of the number of random variables n and the number of training samples N. Thus, it is a scaled version of the likelihood of the training set, which allows us to compare results obtained for a different number of random variables and number of samples. The score is negative and a higher value[8] represents a better result. The goal of our proposals consists of the maximization of this score because it is directly related with Problem 1.

- **Power of generalization**: Measures the capability of the learned models to explain unseen data. The score is computed as the likelihood of a test set (given the model), divided by the product of the number of random variables n and the number of test samples. This is a scaled version of the likelihood in the test data, which is comparable to the power of fitting. Even if this score is not directly related with Problem 1, a high power of generalization is a desirable feature of any probabilistic model. As the number of training and test samples grow, the power of generalization and the power of fitting tend to the same value.

- **Execution time**: We have measured the CPU time required by the learning algorithms. In order to perform a fair comparison, the execution time has been measured for the non-parallelized versions of PFT and SFT, where SFT uses the P&G procedure. The experiments have been carried out in a cluster consisting of Intel-Xeon X5650 at 2.67GHz.

It should be noted that these scores allow us to perform an intuitive scalability study with respect to the number of random variables and the number of training samples. For instance, we can study the convergence of the power of fitting and the power of generalization to the same value as N increases.

PFT and SFT with P&G procedure are compared against the **forward greedy** algorithm (**FG**) adapted from (Desphande et al., 2001). We have implemented an efficient procedure based on the separator-based decomposition, following the intuitions provided in Section 3. The pseudo-code of FG is shown in Algorithm 4. The efficient implementation of FG shares many similarities with the fast implementation proposed by Desphande et al. (2001). At every step, FG adds the candidate edge with the maximum weight (line 5), actualizes the sorted list of candidate edges

[8] i.e., a less negative value

Nomenclature	Name	Nomenclature	Name
PFT	Parallel fractal tree (Alg. 1)	FG	Fast greedy (Alg. 4)
SFT	Sequential fractal free with prune-and-graft (Algs. 2 and 3)	KA	(Kangas et al., 2014)
CL	Chow-Liu algorithm (Chow and Liu, 1968)	SK	(Szántai and Kovács, 2011)

Table 1: The algorithms and their nomenclatures.

(lines 6, 8, 9 and 10), updates the mantles of the separators (lines 6 and 10, see Proposition 19) and updates the set of separators (lines 7, 8 and 9, see Theorem 13). The main difference of PFT and SFT with respect to FG is the way in which they construct an MkDG. PFT and SFT progress by solving separator problems, which are solved by means of a maximum spanning tree solver. In addition, the separator problems can be parallelized using different strategies (see Section 4.5). Finally, PFT and SFT construct a sequence of MiDGs for $i = 2, ..., k$ in $k-1$ growing steps and, at each growing step, the set of separators is fixed.

We have included CL in the comparison as a reference for the following reasons: i) The tree learned by CL is included in the structures obtained by PFT, SFT and FG. Thus, CL is a good reference to measure how much is gained by the three algorithms when the maximum clique size is increased, and ii) we can compare the execution time of the proposed methods with respect to CL, which is the lower bound for the three algorithms.

Additionally, we have compared our proposals with two algorithms of exponential computational complexity: Szantai-Kovac's algorithm (**SK**) (Szántai and Kovács, 2011) and the algorithm presented in (Kangas et al., 2014) (**KA**). It should be highlighted that SK is an approximate algorithm while KA is exact and obtains the optimum solution to Problem 1 (maximum power of fitting). Due to their exponential computational complexity we have restricted the experiments to domains with a low dimensionality ($n = 18$). Table 1 indicates the correspondence between the algorithms and the nomenclature used throughout the experimental section.

Algorithm 4. *(Forward greedy)*
Input: *A data set $\mathcal{D} = \{\boldsymbol{x}^1, ..., \boldsymbol{x}^N\}$, a positive integer k.*
Output: *An MkDG.*
Pseudocode:

1. $\mathcal{S} = \{\emptyset\}$, $\mathcal{E} = \mathbb{E}(\emptyset)$, $\boldsymbol{G} = (V, E)$ where $E = \emptyset$.
2. Compute the weight $I(X_u; X_v|\boldsymbol{X}_S)$ associated to each $\{u, v\} \in \mathcal{E}$.
3. Sort each edge in $\mathcal{E}$ by its corresponding weight, in descending order.
4. For $i = 1, ..., (k-1)n - k(k-1)/2$ do
5. -Add the first edge $\{u, v\}$ from the list $\mathcal{E}$ due to its separator S to E.
6. -Update the mantle of S and remove the edges from $\mathcal{E}$ due to S that are not candidate edges.
7. -If S is not a separator, remove S from $\mathcal{S}$
8. -If needed, add the separator $S \cup \{u\}$ in $\mathcal{S}$, compute the weight associated to its candidate edges, and insert them in $\mathcal{E}$.
9. -If needed, add the separator $S \cup \{v\}$ in $\mathcal{S}$, compute the weight associated to its candidate edges, and insert them in $\mathcal{E}$.
10. -Update the mantle of every $S' \subsetneq S \cup \{u, v\}$, compute the weight of the new candidate edges due to S' and insert them into its sorted list.
11. return $\boldsymbol{G}$

Id	Name	N	n	ID.	Name	N	n
1	KDD: internet usage	10108	66	4	Optical recognition of handwritten digits	5620	65
2	KDD: IPUMS la 99	8844	36	5	SPAM e-mail	4601	58
3	mushroom	8124	22	6	Waveform	5000	41

Table 2: The names of the data sets used in the experimentation, with the number of available samples, N, and the number of implied random variables, n. The data sets will be referred by the identification number (column ID).

6.1. Data from the UCI repository

This section presents a set of results obtained in 6 data sets from the UCI repository with more than 4500 instances. The main features of these domains (n and N) are summarized in Table 2. The continuous random variables have been discretized in two intervals by means of the equal frequency strategy. Next, missing values have been replaced by the most frequent value of the discrete random variable. Finally, random variables with more than 10 states have been removed from the data in order to avoid variables that represent identifiers of the samples.

The experiments have been performed for $k \in \{3, 4, 5\}$ and $N \in \{150, 450, 1350, 4050\}$. For each data set and each value of N we have performed a paired 10 times repeated holdout procedure (Rodríguez et al., 2013) to estimate the power of fitting and the power of generalization of each learned model. In each of the 10 repetitions we have sorted the available data at random and we have selected the first N instances to learn the model and compute its power of fitting. Then, the power of generalization has been computed using the instances from position 4051. Next, we have computed estimated power of fitting and power of generalization using the average across the 10 repetitions. The experiments include CL, which represents the optimal solution to Problem 1 for $k = 2$. The first growing step of PFT and SFT is equivalent to CL and, thus, CL is used in order to show the relative increase of the power of fitting as k increases.

Two different experiments have been performed. In the first, we compare PFT, SFT, FG, KA and SK. In order to include SK and KA, we have selected the 18 variables with higher entropy values. This experiment is used to illustrate the behavior of PFT and SFT against the optimum solution of instances of Problem 1, which are given by KA. Figures 9 and 10 summarize the obtained average estimated power of fitting and power of generalization values across the six data sets from the UCI repository. The results clearly indicate that PFT, SFT, FG, KA and SK obtain similar results in terms of the power of fitting and power of generalization, KA being slightly better in terms of power of fitting and PFT being slightly better in terms of power of generalization for $k \in \{4, 5\}$. On the one hand, the obtained results illustrate that as N increases, the power of fitting and power of generalization converge to the same values. For $k = 3$ the convergence is attained for $N = 4050$. On the other hand, even with $N = 4050$ there is a discrepancy between the power of fitting and power of generalization for $k = 4$ and (especially) for $k = 5$, due to the overfitting phenomena.

In the second experiment, we compare PFT, SFT and FG using all the variables (see Table 2 for further details). Figures 11 and 12 summarize the obtained results. The conclusions are similar to the previous experiment. All the learned models obtain similar results. As N increases, the power of fitting and power of generalization converge to the same value. The convergence is reached for $k = 2$ (CL) with $N = 1350$ and for $k = 3$ with $N = 4050$. For $k = 4$ and $k = 5$ more data seems to be required to reach the convergence, which indicates that there is still room for improvement regarding the power of generalization.

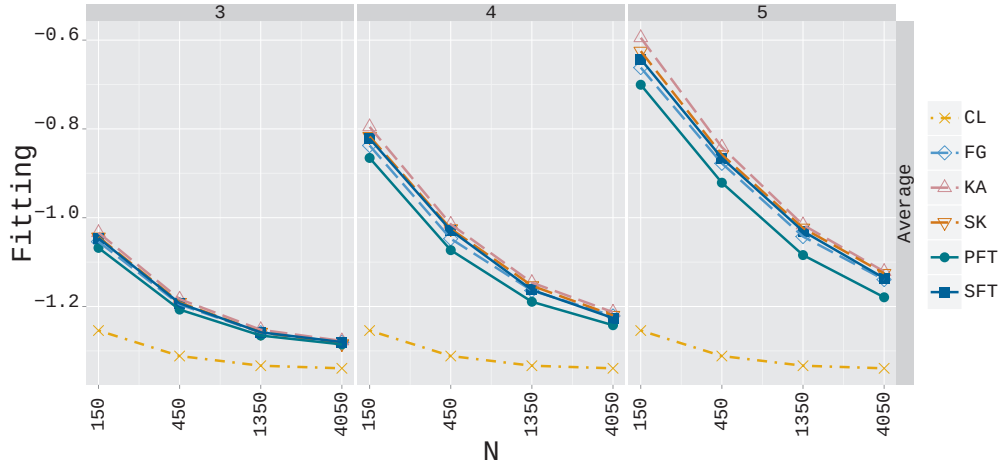


Figure 9: The figure shows the evolution of the average power of fitting with respect to $N \in \{150, 450, 1350, 4050\}$ for $k \in \{3, 4, 5\}$ in the real data using $n = 18$ variables. KA represents the optimal solution to Problem 1.

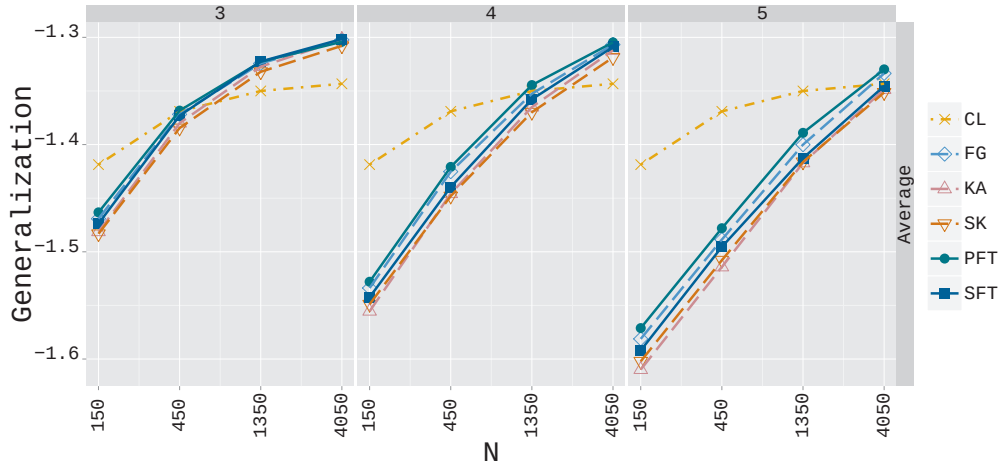


Figure 10: The figure shows the evolution of the average power of generalization with respect to $N \in \{150, 450, 1350, 4050\}$ for $k \in \{3, 4, 5\}$ in the real data using $n = 18$ variables.

6.2. Artificial domains with MkDG structure

This section presents a set of results obtained in domains[9] with MkDG structure. It should be noted that the generated MkDG structures represent the optimal solution to Problem 1 given enough data. By means of these domains we can study the error produced by our approaches with respect to the optimum solutions of instances from an NP-hard problem with more than 1000 variables. This analysis allows us to perform a scalability study with respect to N, n and

[9] i.e., probability distributions

k. In this section we study the evolution of the power of fitting and power of generalization with respect to N until the convergence is reached for different values of n and k.

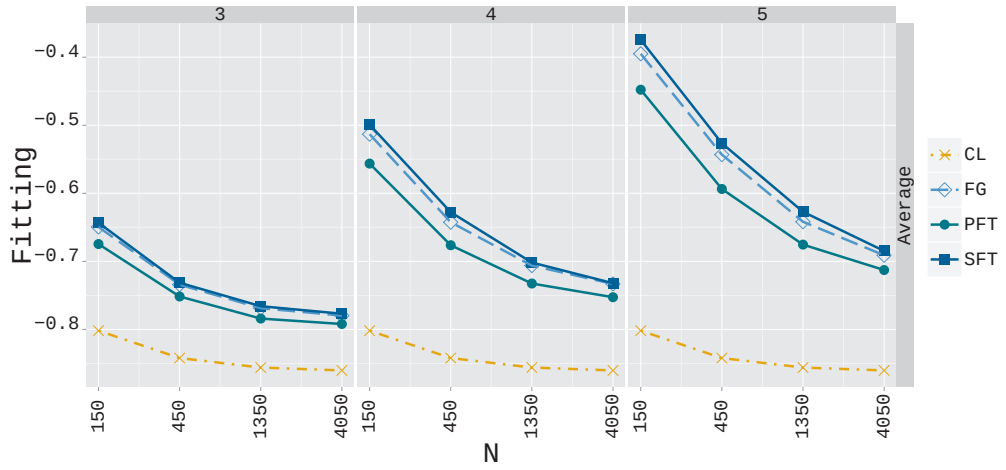


Figure 11: The figure shows the evolution of the average power of fitting with respect to $N \in \{150, 450, 1350, 4050\}$ for $k \in \{3, 4, 5\}$ in the real data. KA represents the optimal solution to Problem 1.

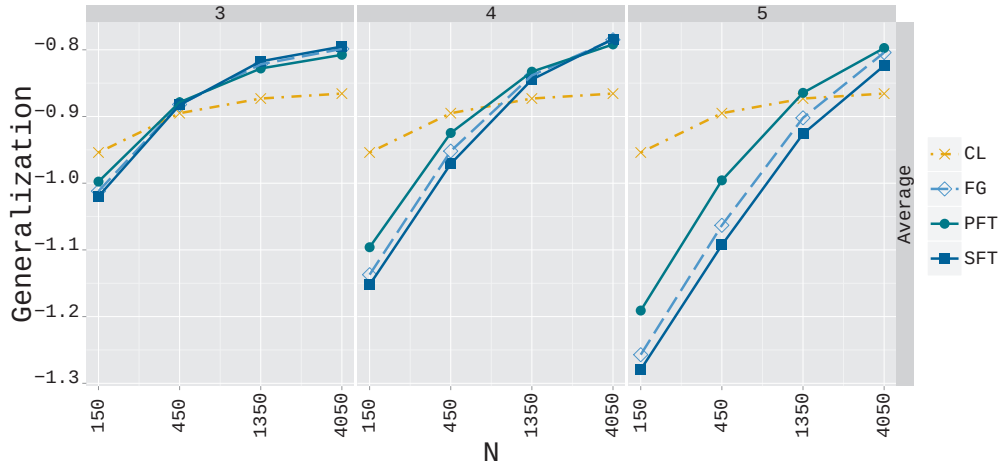


Figure 12: The figure shows the evolution of the average power of generalization with respect to $N \in \{150, 450, 1350, 4050\}$ for $k \in \{3, 4, 5\}$ in the real data.

These structures have been randomly obtained using the procedure described in Algorithm 5. We have generated structures for $k \in \{3, 4, 5\}$ and $n \in \{40, 80, 160, 320, 640, 1280\}$. In addition, in order to compare PFT and SFT against SK and KA, we have generated domains with 18 variables. The random variables are binary and the parameters of the model associated to each structure have been randomly sampled from a symmetric Dirichlet distribution with $\alpha = 1$. For each type of domain, we have generated at random 50 different models (structure and

parameters). In order to study the influence of the amount of available data, we have sampled training sets with different sizes, $N \in \{100, 300, 1000, 3000, 10000, 30000\}$. The generated test sets used for computing the power of generalization are of size $M = 10000$. All the figures that summarize the results obtained in artificial domains with an MkDG structure include the power of generalization of the true model, which represents the optimal power of generalization value and, hence, the optimal solution to Problem 1 given enough data.

In order to perform a correct interpretation of the obtained results, we discuss the effect of fixing Dirichlet's $\alpha = 1$ value to sample the parameters of the artificial domains:

- Let C be a clique of the generated MkDG. The value of $\alpha = 1$ indicates that all the parameters associated to the marginal distribution $p(\boldsymbol{X}_C)$ have the same probabilities of being randomly sampled from the Dirichlet distribution. In order words, it has no preference towards any type of marginal distribution $p(\boldsymbol{X}_C)$.

- A fixed value for α allows us to obtain, on average, probability distributions with similar power of fitting (and generalization) for the different values of k and n. In other words, on average, the true model obtains similar power of fitting (and generalization) values for different configurations of k and n. Thus, the power of fitting (and generalization) values obtained by an algorithm can be considered comparable for different settings, which allows to perform scalability studies with respect to k and n highlighting its relative merits and defects.

- A fixed value has consequences in the difficulty of the problem. Let S be a subset of C. By the aggregation property of the Dirichlet distribution, the marginal distribution $p(\boldsymbol{X}_S)$ can be interpreted as having been generated using a Dirichlet distribution with parameter $\alpha = 2^{|C \setminus S|}$. This parameter favors, as k increases, the sampling of more uniform low order distributions which tends to produce smaller mutual information quantities. In other words, as k increases, the low order marginal distributions tend to be less informative. Thus, the problem of learning the MkDG becomes more difficult as k increases: more edges must be learned using less informative marginal low dimensional distributions.

Figures 13 and 14 summarize the power of fitting and power of generalization obtained with SK, KA, PFT and SFT with $n = 18$. SFT can obtain a power of fitting close to the optimum, given by KA, and similar to SK. This fact highlights that the proposed approach is an effective alternative to deal with Problem 1. In addition, SFT obtains a power of generalization similar to SK and close to KA. In this experiment we can observe the converge of the power of fitting and power of generalization for the different k values.

Figure 15 and 16 show a subset of the obtained results ($n \in \{80, 320, 1280\}$) with FG, PFT and SFT, for the sake of brevity. The presented results highlight the main trends observed in the whole experimentation. The following conclusions can be drawn from the results:

- SFT obtains the best power of fitting results for all configurations of n, N and k. The average second best results are obtained by FG and the worst results by PFT.

- SFT obtains the best power of generalization results in most configurations of n, N and k. In additional experiments not shown in this work, we have observed that PFT obtains the best power of generalization for $N = 30$, especially for $n = 1280$.

- As k increases, the differences among the models increase, while the ranking among them is maintained.

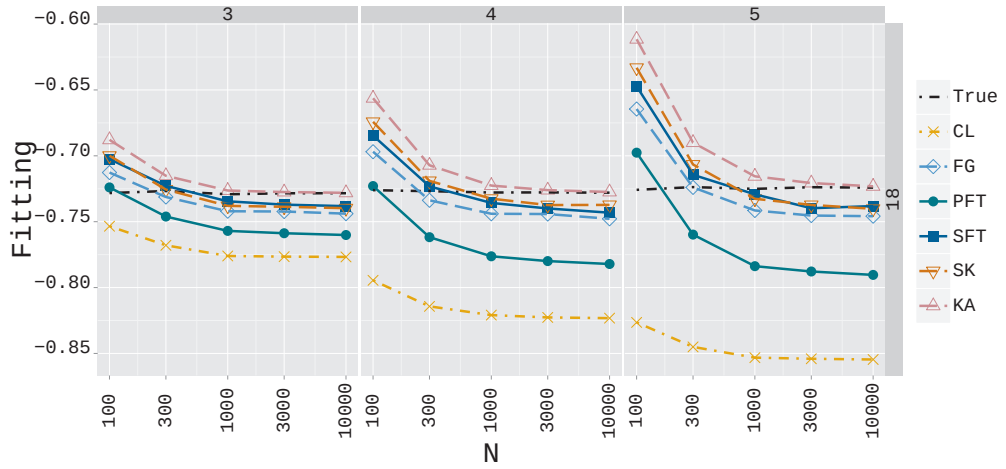


Figure 13: The evolution of the power of fitting with respect to $N \in \{100, 300, 1000, 3000, 10000\}$ for $n = 18$ in domains with MkDG structure. KA represent the optimal solution to Problem 1.

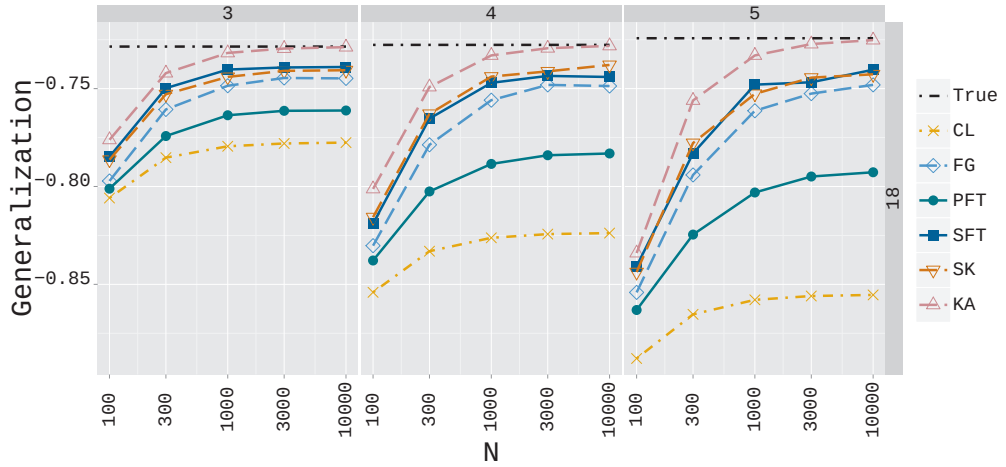


Figure 14: The evolution of the power of generalization with respect to $N \in \{100, 300, 1000, 3000, 10000\}$ for $n = 18$ in domains with MkDG structure.

- As n increases, SFT, FG and PFT approaches tend to obtain slightly worse results compared with the true model, which indicates a good scalability with respect to n. As k increases, the worsening is higher. This is a direct consequence of the choice of the parameters for the Dirichlet distributions, which creates harder problems as k increases. The worsening of PFT with respect to k is higher, mainly due to the chain caveat.

- As n increases the differences in terms of power of fitting and power of generalization between PFT and SFT become smaller. This trend suggests that PFT could reach the power of fitting and the power of generalization values of SFT in MkDG domains with higher dimensionality.

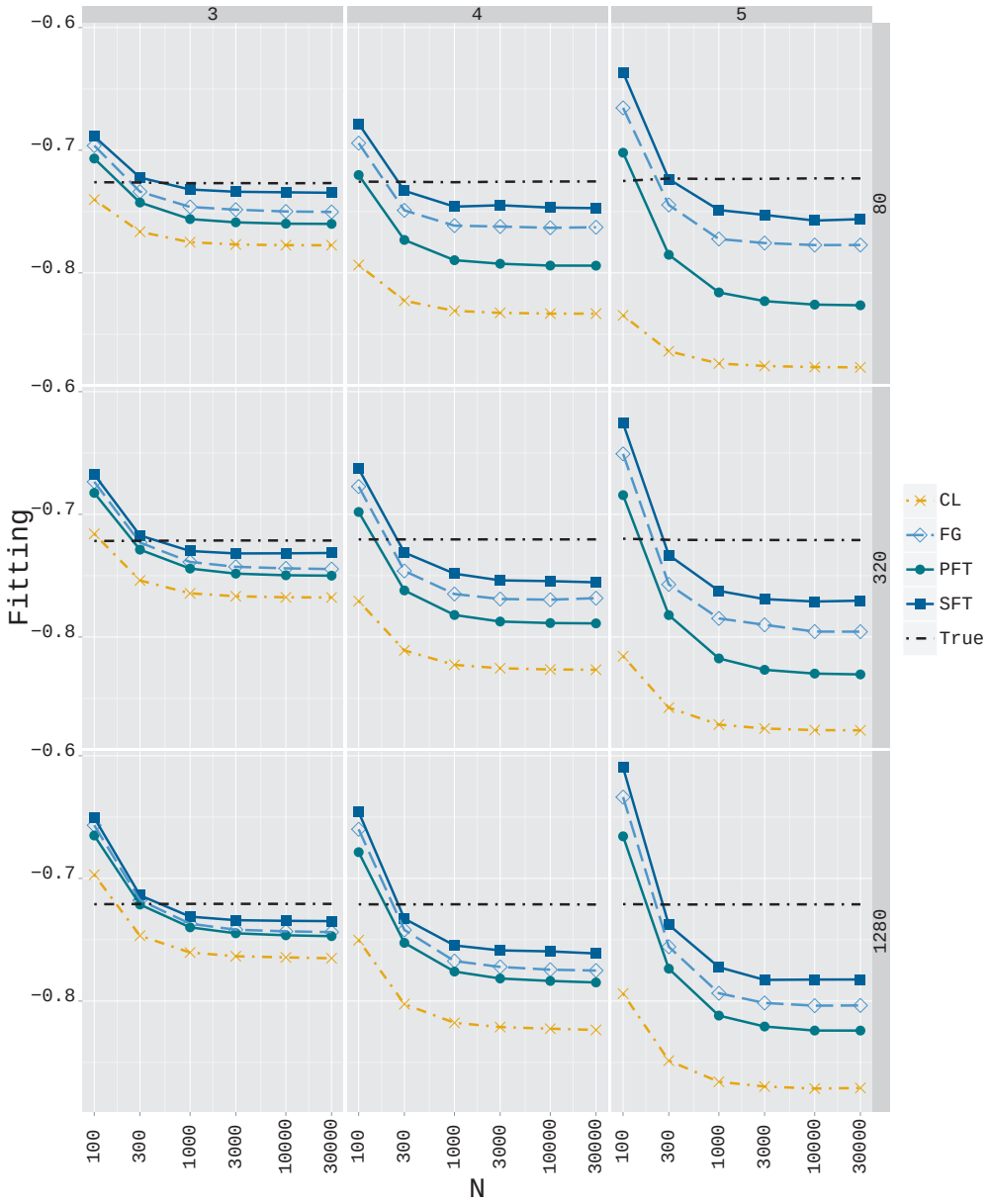


Figure 15: The evolution of the power of fitting with respect to $N \in \{100, 300, 1000, 3000, 10000, 30000\}$ for $n \in \{80, 320, 1280\}$ in domains with MkDG structure.

6.3. Artificial domains without MkDG structure

This section presents results in domains with a sDG structure with a maximum clique size $s = 10$. The structures of the randomly generated decomposable models have been generated using the pseudocode shown in Algorithm 6 in Appendix B. We want to study the effects of the

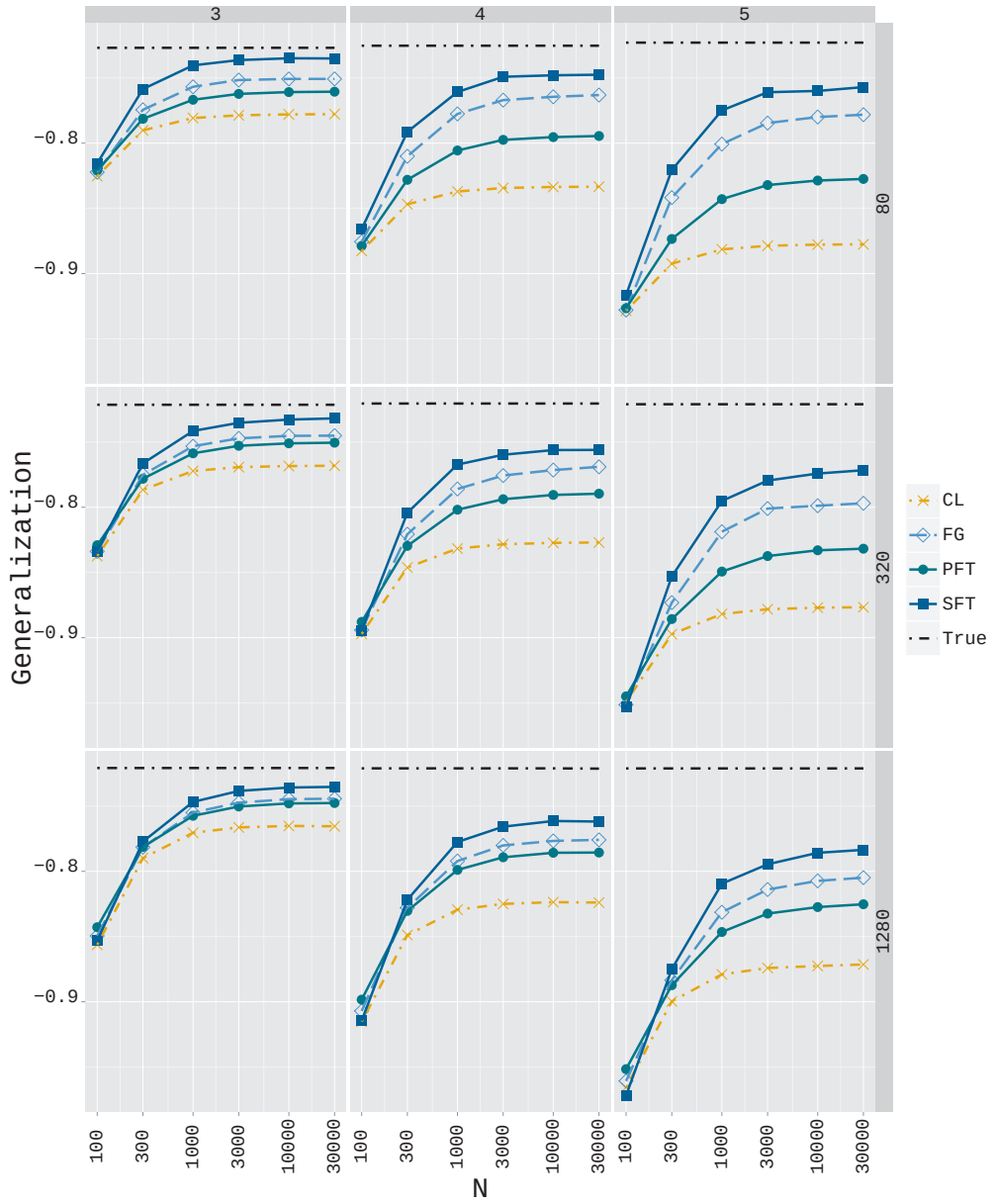


Figure 16: The evolution of the power of generalization with respect to $N \in \{100, 300, 1000, 3000, 10000, 30000\}$ for $n \in \{80, 320, 1280\}$ in domains with MkDG structure.

value of k of the number of variables n and the size of the training set N in domains with a maximum clique size higher than $K \in \{3, 4, 5\}$. The experiments have been performed for the same algorithms and using the parameters described at Section 6.2

The comparison of PFT and SFT against SK and KA in domains with $n = 18$ random vari-

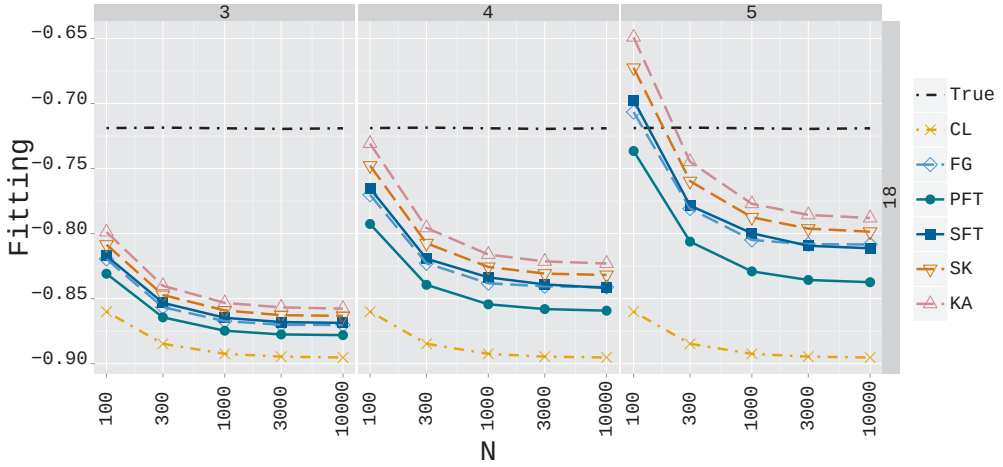


Figure 17: The evolution of the power of fitting with respect to $N \in \{100, 300, 1000, 3000, 10000\}$ for $n = 18$ in domains without MkDG structure. KA represent the optimal solution to Problem 1.

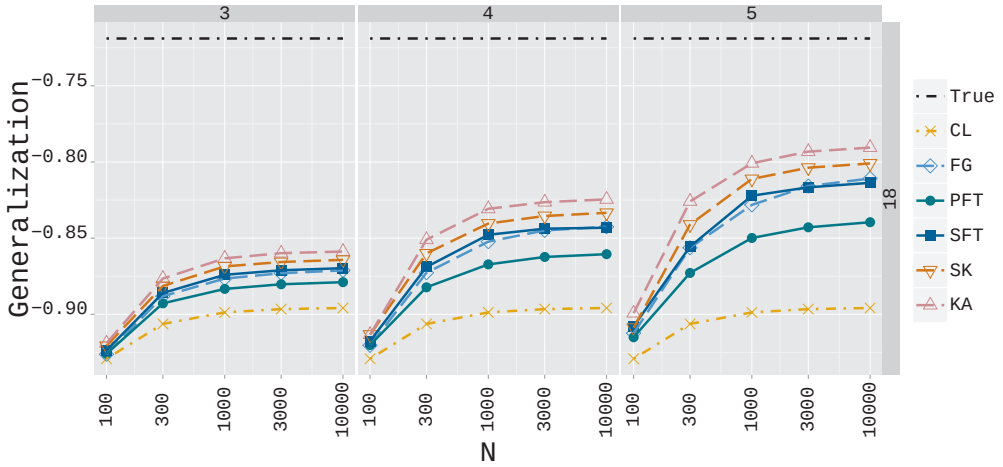


Figure 18: The evolution of the power of generalization with respect to $N \in \{100, 300, 1000, 3000, 10000\}$ for $n = 18$ in domains without MkDG structure.

ables is summarized in Figures 17 and 18. In this experiment we can observe that the optimum MkDG, given by KA, has obtained a worse result compared with the true model than in the experiments summarized in Figures 14 and 13. This is a direct consequence of s being greater than k and of the the Dirichlet aggregation property: as s increases the marginals of order k tend to be more uniform. In consequence, models with MkDG structure tends to obtain worse power of fitting values than the true model due to its lower maximum clique size k. Additionally, due to the same property, the differences among the different models tend to decrease because the weight associated to the candidate edges is lower. SFT and FG obtain results similar to SK, which are close to the optimum. As k increases the models obtain better power of generalization

values given enough data.

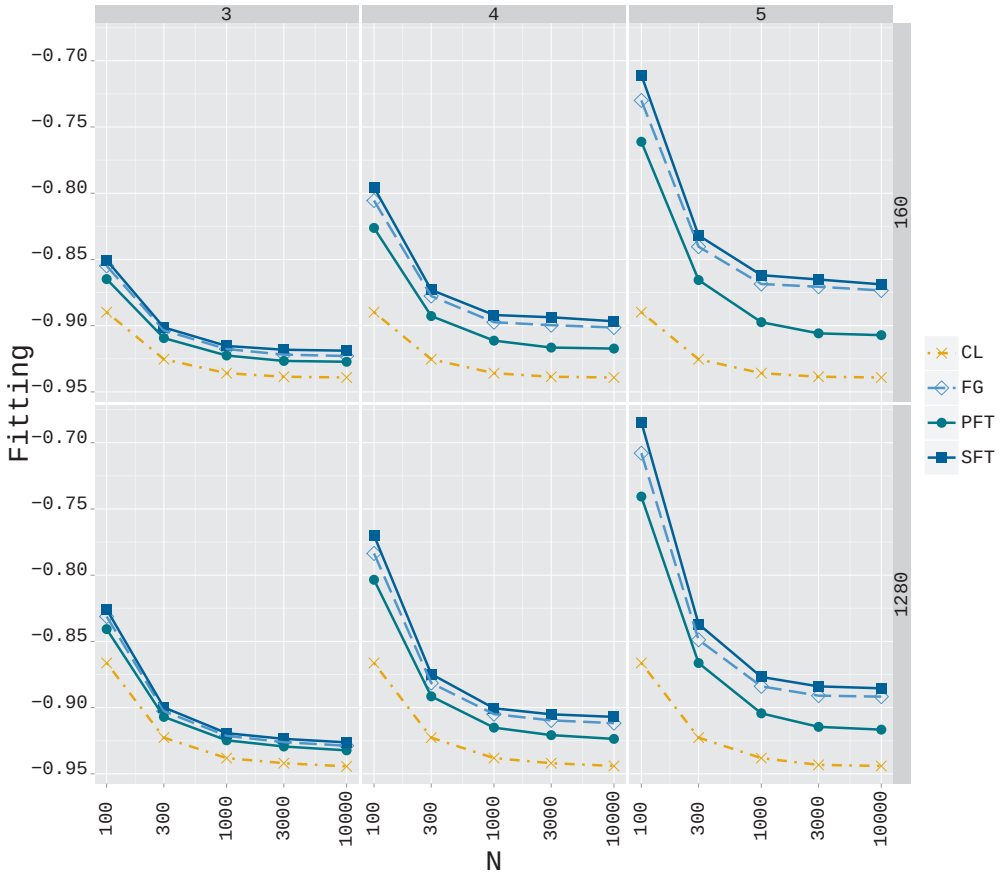


Figure 19: The evolution of the power of fitting with respect to $N \in \{100, 300, 1000, 3000, 10000\}$ for $n \in 160, 1280$ in domains without MkDG structure.

Figures 19 and 20 summarize the results obtained with CL, FG, PFT and SFT for $n \in \{160, 1280\}$, for the sake of brevity. The experiments show similar trends with $n \in \{80, 160, 320, 640, 1280\}$. For $k = 3$, FG, PFT and SFT obtain similar results. SFT and FG obtain quite similar results in terms of power of fitting and generalization for $k \in \{4, 5\}$, SFT being slightly better. As k increases, PFT obtains worse results compared to SFT and FG due to the combined effect of the chain caveat and the uniformity of the low order marginal distributions. As n increases, PFT obtains closer results to SFT and FG. This trend is also observed in the domains with an MkDG structure.

The execution times for different settings of k, n and N are summarized in Figures 21 and 22. These values have been obtained with a non-parallel implementation of PFT and SFT. PFT is the most efficient algorithm for all the configurations of k, n and N. It provides times which are very close to CL. As k increases, the execution time slightly increases for PFT. SFT is slower than FG in most of the configurations of n, N and k, because it considers more candidate edges and thus it computes a higher number of mutual information quantities. However, for high dimensional

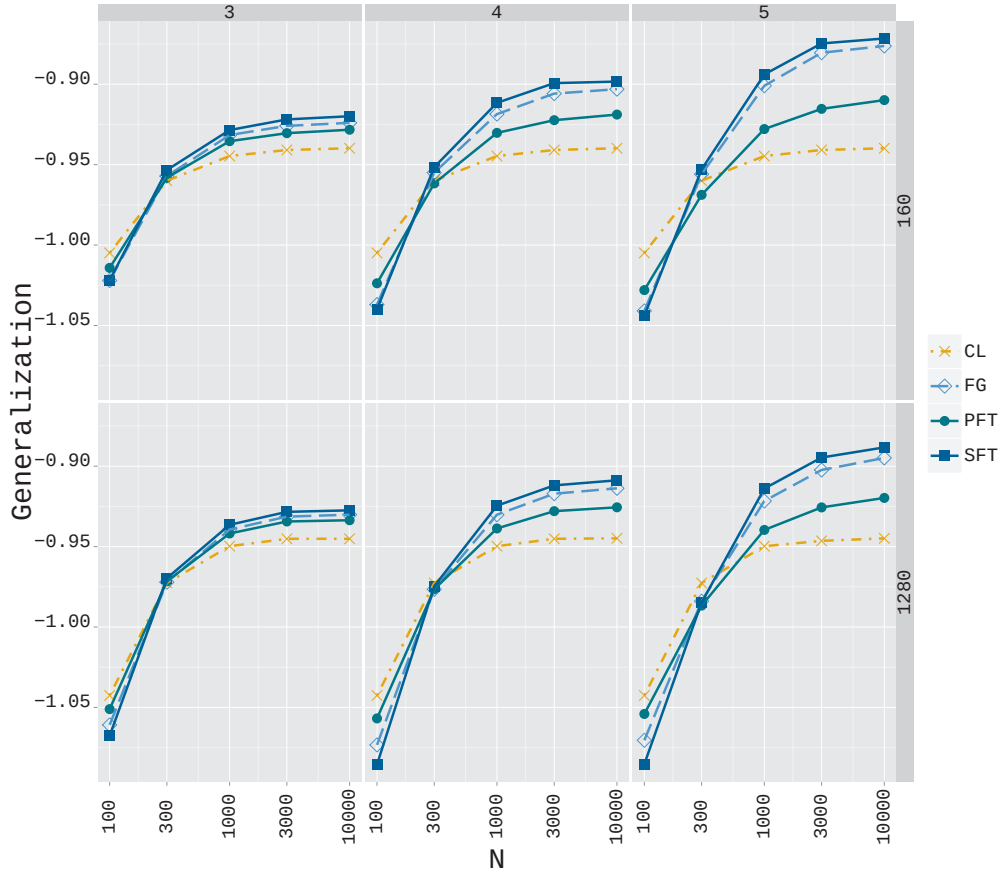


Figure 20: The evolution of the power of generalization with respect to $N \in \{100, 300, 1000, 3000, 10000\}$ for $n \in \{160, 1280\}$ in domains without MkDG structure.

domains, given relatively moderate training data (e.g., $n = 2560$ and $N < 3000$), the time consumed by FG is higher than the time required by SFT for computing the mutual information quantities due to its computational complexity, which is cubic in n. It should be highlighted that this effect tends to increase as n increases, since the worst-case computational complexity of SFT is quadratic in n. We have observed that, for SFT, most of the execution time is consumed by the P&G procedure. In additional experimentation (not shown in this work), SFT without the P&G operator obtains similar execution times and better power of fitting than PFT.

6.4. Practical suggestion for determining the parameter k.

This work is focused on learning decomposable models with an MkDG structure using constructive procedures guided by the likelihood of the available data (Problem 1). In other words, the proposed methods are focused on the maximization of the power of fitting given a value k, which controls the complexity of the learned models. However, in most of the real world applications, we are interested in models that maximize the likelihood of unseen data, that is, in the maximization of the power of generalization.

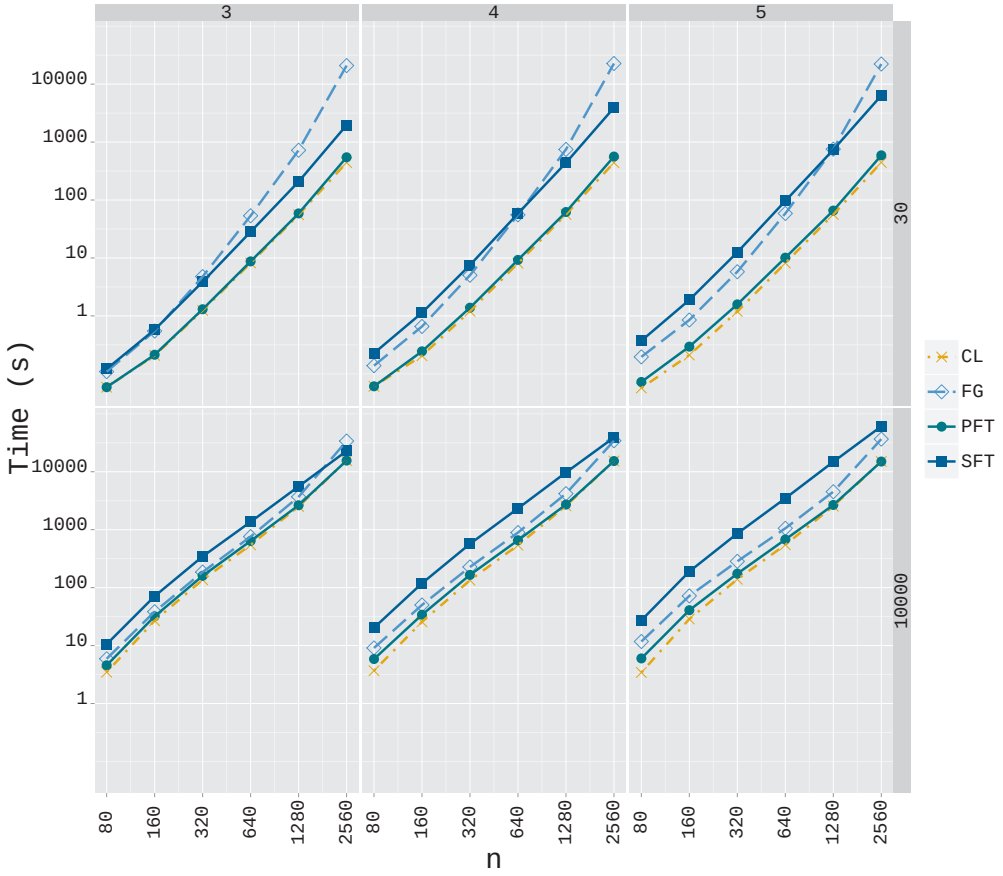


Figure 21: The evolution of the computational time (in seconds) with respect to $n \in \{80, 160, 320, 640, 1280, 2560\}$ for $N \in \{30, 10000\}$.

As we have seen in the performed experiments, often the size of the training data N can have dramatic effects on the power of generalization and the power of fitting values obtained by MkDGs for different values of k. Usually, on average, as the size of the training data N increases, the power of fitting of the learned model decreases, while its power of generalization increases. For a fixed value k both measures converge to the same value given enough data and this value tends to be higher as the value of k increases. Unfortunately, as k increases, the convergence tends to be reached with higher values of N. In real world applications we often have a limited amount of data and, thus, in order to maximize the power of generalization of the learned model, we should select k carefully taking into account the particularities of the available data. We recommend selecting the highest value of k for which the estimated power of fitting and the estimated power of generalization have similar values.

PFT and SFT have two important features that ease the selection of a good value of the parameter k: i) They have a low computational complexity in the worst case and they can be parallelized, and ii) they produce a sequence of MiDGs for $i = 2, ..., k$. Both features allow us to select an appropriate value of k by using computational intensive estimates of the power of

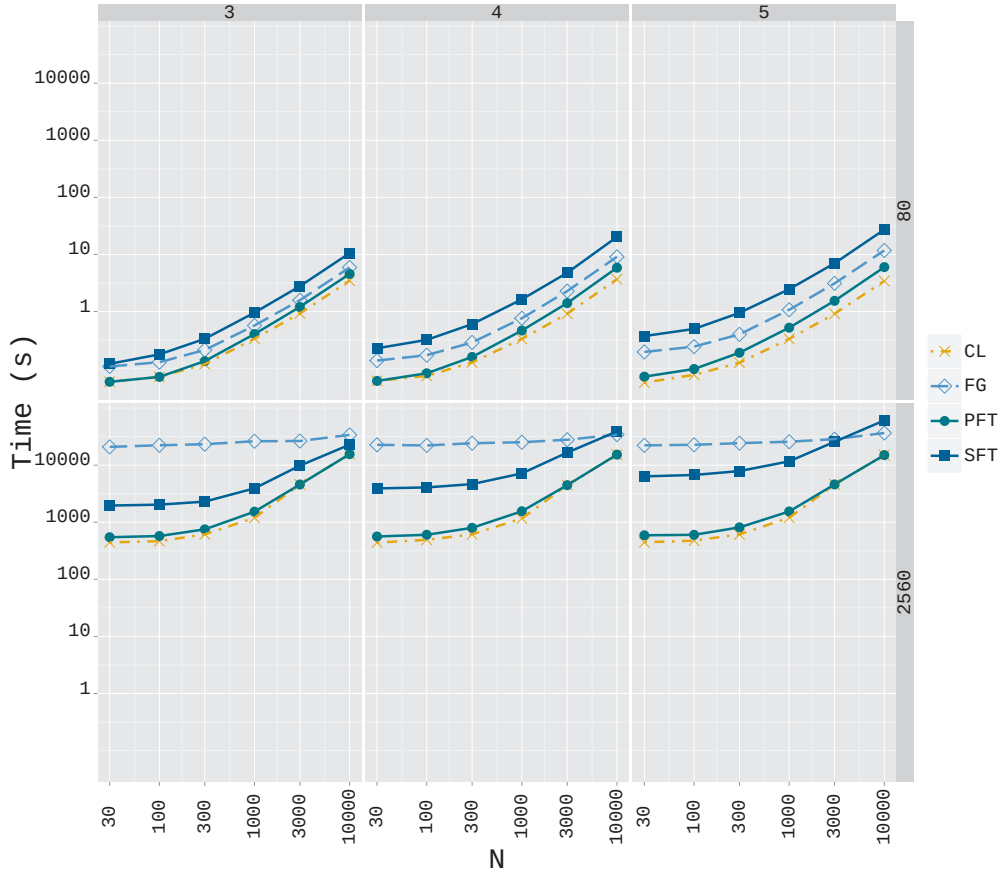


Figure 22: The evolution of the computational time (in seconds) with respect to $N \in \{30, 100, 300, 1000, 3000, 10000\}$ for $n \in \{80, 2560\}$.

generalization, such as repeated holdout, cross-validation and bootstrap among others (Rodríguez et al., 2013). All of the mentioned estimators take a subset of the available data for learning and then the learned model is tested on the unobserved data. The process is repeated several times[10] and then the estimate of the power of generalization is computed by averaging the obtained values. With PFT and SFT two different strategies can be used, among others. First, it is possible to construct an entire sequence of MiDGs for $i = 2, .., k$, where k can be selected taking into account the the size of the available data and the available computational resources. Then, we can estimate the power of generalization of the entire sequence of models and select the parameter i that maximizes the power of generalization. Secondly, it is possible to estimate the power of generalization at each growing step and use a stop criterion based on the power of generalization to determine a good value for the parameter k.

[10] As the number of repetitions increases, the variance of the estimator is reduced (Rodríguez et al., 2013)

7. Conclusions

Learning maximum likelihood decomposable models with a maximum clique size of k (Problem 1) is known to be an NP-hard problem for $k > 2$. This problem can be solved by learning a maximal k-order decomposable graph (MkDG).

In this work we have presented the family of the fractal tree algorithms[11]. These algorithms are focused on learning MkDGs for dealing with Problem 1 and they have a computational complexity of $\mathcal{O}(k \cdot n^2)$. They are based on a divide-and-conquer strategy. Our approaches divide Problem 1 into $k-1$ subproblems (see Problem 2), which produces a sequence of MiDGs for $i = 2, .., k$, each coarser than the previous one. Additionally, Problem 2 is decomposed in terms of the separators of the input MiDG (see Problem 3). The subproblems associated to the separators are efficiently solved using the novel generalized Chow-Liu algorithm. We have proposed two particular fractal tree algorithms: parallel fractal tree (PFT) and sequential fractal tree (SFT). PFT solves the separators at each growing step in parallel, while SFT solves them sequentially taking into account their interactions. In addition, a prune-and-graft procedure specifically designed for MkDGs has been proposed. This operator increases the mobility of the leaf vertices and can be used after each growing step of the fractal tree algorithms. In the experimental section we have measured the power of fitting and the power of generalization using artificial and real domains. Sequential fractal tree with the prune-and-graft procedure has shown a better power of fitting and similar power of generalization than PFT (see Section 6). The fractal tree family of algorithms has shown a competitive behavior for dealing with Problem 1.

All the provided procedures are efficient, modular and can be parallelized using the decomposition of an MkDGs in terms of its separators. Additionally, the computation of the weights required to solve each separator problem can be parallelized following the procedure proposed in (Madsen et al., 2014). Last but not least, they produce a sequence of MiDGs for $i = 2, ..., k$ which allow us to select the most suitable model for the problem at hand. Due to these features, we consider that the proposed algorithms are especially suitable for modeling high-dimensional domains, which is a central aspect of the massive data analysis (Jordan et al., 2013).

Acknowledgements

A. Pérez is supported by the Basque Government through the BERC 2014-2017 program and by Spanish Ministry of Economy and Competitiveness MINECO: BCAM Severo Ochoa excellence accreditation SEV-2013-0323. I. Inza and J. A. Lozano are partially supported by the Saiotek and IT609-13 programs (Basque Government), TIN2013-41272-P (Spanish Ministry of Science and Innovation) and COMBIOMED network in computational bio-medicine (Carlos III Health Institute).

Appendix A: Theoretical results

In this appendix we prove the following results: i) The addition of a candidate edge $\{u, v\}$ due to a separator S decreases the entropy of a decomposable model in $\hat{I}(X_u; X_v|\boldsymbol{X}_S)$, ii) Problem 1 can be solved constructing a sequence of coarser MiDGs, iii) Problem 2 is NP hard, iv) we

[11] The implementation in Python of the proposed algorithms is publicly available at https://bitbucket.org/AritzPerez/fractaltree.

Table 3: Main concepts

SYMBOL	CONCEPT	REFERENCE
$\boldsymbol{G}[S]$	Subgraph induced by the set of vertices S	Sec. 2.1
$\boldsymbol{G} \prec \boldsymbol{G}^+$	$\boldsymbol{G}^+$ is coarser than $\boldsymbol{G}$	Def. 1, Sec. 2.1
$\mathbb{N}_{\boldsymbol{G}}(S)$	Common neighbor of S	Def. 2, Sec. 2.1
$\boldsymbol{G}[\mathbb{N}(S)]$	Mantle of S	Def. 2, Sec. 2.1
$\mathbb{C}(\boldsymbol{G})$	Set of cliques of $\boldsymbol{G}$	Sec. 2.1
$\mathbb{S}(\boldsymbol{G})$	Set of minimal separators of $\boldsymbol{G}$	Sec. 2.1
$\mathbb{E}_k(\boldsymbol{G})$	Candidate edges that create cliques of size k	Def. 4, Sec. 2.1
DG	Decomposable graph	Sec. 2.1
MkDG, $\boldsymbol{G}_k$	Maximal k-order decomposable graph	Def. 5, Sec. 2.1
$(k+1)$DG	$(k+1)$-order decomposable graph	Def. 5, Sec. 2.1
d_S	Degree of the separator S	Sec. 2.2
$p_{\boldsymbol{G}}$	Decomposable model associated to $\boldsymbol{G}$	Sec. 2.2
$\mathbb{E}_{\boldsymbol{G}}(S)$	Candidate edges due to S in $\boldsymbol{G}$	Def. 7, Sec. 3.1

characterize all the changes that produce the addition of a candidate edge to the set of cliques and separators of a DG, v) all the separators of size $k-1$ of a DG coarser than a given MkDG are included in the set of separators of the MkDG, vi) any candidate edge that can be added to a DG coarser than an MkDG, which creates a clique of size $k+1$, is a candidate edge due to a separator of the MkDG, vii) the justification of the separator-based decomposition introduced in Section 3.1, viii) the addition of an edge to a separator can only modify the mantle of an adjacent separator, ix) the number of the mutual information quantities computed by a growing step of PFT and SFT is $\mathcal{O}(n^2)$ in the worst case, which corresponds to an MkDG with a single separator x) the i^{th} growing step of the PFT and SFT always produces an M$(i+1)$DG and, xi) P&G procedure constructs an MkDG given an input MkDG with an equal or lower entropy (higher or equal likelihood). Table 3 summarizes the main concepts required to understand the provided results.

We start by determining the weight used by GCL.

Proposition 10. *Let $\boldsymbol{G} = (V, E)$ be a DG where S is a minimal separator and $\{u, v\} \in \mathbb{E}(S)$. Let $\boldsymbol{G}^+ = (V, E \cup \{u, v\})$ be the DG obtained by adding $\{u, v\}$ to $\boldsymbol{G}$, and let $\mathcal{D}$ be the data set used to learn the maximum likelihood parameters of the decomposable models associated to $\boldsymbol{G}$ and $\boldsymbol{G}^+$. The next equality is verified*

$$\hat{H}(\boldsymbol{G}) - \hat{H}(\boldsymbol{G}^+) = \hat{I}(X_u; X_v | \boldsymbol{X}_S) \tag{2}$$

where $\hat{H}(\boldsymbol{G}) = \frac{1}{N} \sum_{\boldsymbol{x} \in \mathcal{D}} p_{\boldsymbol{G}}(\boldsymbol{x})$ is the empirical entropy associated to $\boldsymbol{G}$ and $\hat{I}(X_u; X_v | \boldsymbol{X}_S) = \frac{1}{N} \sum_{\boldsymbol{x} \in \mathcal{D}} \frac{\hat{p}(x_u, x_v | \boldsymbol{x}_S)}{\hat{p}(x_u | \boldsymbol{x}_S) \hat{p}(x_v | \boldsymbol{x}_S)}$ is the empirical conditional mutual information.

Proof. The result is directly given by the equality $\frac{p_{\boldsymbol{G}^+}(\boldsymbol{x})}{p_{\boldsymbol{G}}(\boldsymbol{x})} = \frac{\hat{p}(x_u, x_v | \boldsymbol{x}_S)}{\hat{p}(x_u | \boldsymbol{x}_S) \hat{p}(x_v | \boldsymbol{x}_S)}$ for any $\boldsymbol{x}$ □

The next result indicates that Problem 1 can be solved by constructing a sequence of coarser MiDGs, for $i = 2, ..., k$.

Proposition 11 (Lemma 2.21, Lauritzen (1996))**.** *Given a DG, it is possible to construct any coarser DG by adding a sequence of candidate edges.*

This result states that, without loss of generality, any coarser DG can be obtained by constructing a sequence of coarser DGs which differ exactly in one candidate edge. As a direct consequence, for any MkDG $\boldsymbol{G}_k$ there is at least a sequence of MiDG structures, $\boldsymbol{G}_i$ for $i = 2, ..., k$, starting from the empty graph, where $\boldsymbol{G}_2 \prec \boldsymbol{G}_3 \prec ... \prec \boldsymbol{G}_{k-1} \prec \boldsymbol{G}_k$. According to this result, Problem 1 could be solved by solving a sequence of Problems 2.

Unfortunately, Problem 2 is still NP-hard and we propose an efficient approach to this problem based on the separator-based decomposition described in Section 3.1.

Corollary 12. *Problem 2 is NP-hard for $k > 1$.*

Proof. The proof is equivalent to the proofs for Problem 1 provided in Srebro (2000), Theorem 4.1, and in Srebro (2003), Corollary 3. □

Next, we describe the modifications that are produced by the addition of a candidate edge to a decomposable graph in terms its cliques and separators. This result is closely related to Theorem 4.2 in (Desphande et al., 2001).

Theorem 13. *Let $\boldsymbol{G}$ be a DG with the candidate edge $\{u, v\}$ due to S and let $\boldsymbol{G}^+$ be the DG obtained from $\boldsymbol{G}$ by adding $\{u, v\}$. $\boldsymbol{G}^+$ fulfills the following properties:*

i) $\mathbb{C}(\boldsymbol{G}^+) \setminus \mathbb{C}(\boldsymbol{G}) = \{S \cup \{u, v\}\}$
ii) $\mathbb{C}(\boldsymbol{G}) \setminus \mathbb{C}(\boldsymbol{G}^+) \subseteq \{S \cup \{u\}, S \cup \{v\}\}$
iii) $\mathbb{S}(\boldsymbol{G}^+) \setminus \mathbb{S}(\boldsymbol{G}) \subseteq \{S \cup \{u\}, S \cup \{v\}\}$
iv) $\mathbb{S}(\boldsymbol{G}) \setminus \mathbb{S}(\boldsymbol{G}^+) \subseteq \{S\}$

where $\{u, v\}$ is a candidate edge due to S in $\boldsymbol{G}$.

Proof. First, it should be noted that, by Theorem 4, there exists a set of vertices S that is the minimal separator of u and v where $\{u, v\} \subseteq \mathbb{N}_{\boldsymbol{G}}(S)$. Thus, there exists a chain of cliques $C_1, ..., C_i, C_{i+1}, ...C_m$ for the DG $\boldsymbol{G}$, where $C_i \cap C_{i+1} = S$, $S \cup \{u\} \subseteq C_i$ and $S \cup \{v\} \subseteq C_{i+1}$.

Claim (i) is directly given in Theorem 4. From here on we denote $S \cup \{u, v\}$ by C.

By claim (i), $\mathbb{C}(\boldsymbol{G}^+)$ can not contain any proper subset of $S \cup \{u, v\}$. Since C_i and C_{i+1} are cliques for $\boldsymbol{G}$, then $\mathbb{C}(\boldsymbol{G})$ can not contain any proper subset of $S \cup \{u\}$ and $S \cup \{v\}$. Besides, $\mathbb{C}(\boldsymbol{G})$ can not contain any superset of $\{u, v\}$ because the edge $\{u, v\}$ is not included in $\boldsymbol{G}$. Therefore, the unique maximal cliques that can be removed from $\mathbb{C}(\boldsymbol{G})$ by the addition of $\{u, v\}$ to $\boldsymbol{G}$ are $S \cup \{u\}$ and $S \cup \{v\}$, which proves the claim (ii).

By claims (i) and (ii), we can conclude that it is possible to define one of the next chains of cliques for $\boldsymbol{G}^+$: I) $C_1, ..., C_i, C, C_{i+1}, ...C_m$ if and only if $S \cup \{u\} \subsetneq C_i$ and $S \cup \{v\} \subsetneq C_{i+1}$; II) $C_1, ..., C_{i-1}, C, C_{i+1}, ...C_m$ if and only if $S \cup \{u\} = C_i$ and $S \cup \{v\} \subsetneq C_{i+1}$; III) $C_1, ..., C_i, C, C_{i+2}, ...C_m$ if and only if $S \cup \{u\} \subsetneq C_i$ and $S \cup \{v\} = C_{i+1}$; and IV) $C_1, ..., C_{i-1}, C, C_{i+2}, ...C_m$ if and only if $S \cup \{u\} = C_i$ and $S \cup \{v\} = C_{i+1}$. Therefore, the changes in the sets of separators of $\boldsymbol{G}$ due to the addition of $\{u, v\}$ are given by the intersections of C_i, C and C_{i+1}. In chains I) and III) the separator $S \cup \{u\}$ appears in the list of separators associated and in chains I) and II) the separator $S \cup \{v\}$ appears in the list of separators associated, which proves claim (ii). In the four chains of cliques that can be generated for $\boldsymbol{G}^+$, the times that the separator S appears in its associated list of separators is reduced in one, which proves claim (iv). □

The first claim describes the clique that is created after the addition of a candidate edge, the second the cliques that could be removed from the coarser structure, the third determines the separators that could be produced and the fourth the separators that could be removed.

Since the separators are a central aspect of FT, we now introduce a result that describes the separators that can be found in a DG coarser than a given DG.

Corollary 14. *Let $\boldsymbol{G}$ be a DG and $\boldsymbol{G}^+$ be a coarser DG, then the set of separators of $\boldsymbol{G}^+$ is a subset of the set $\{S^+ : \text{exists } S \in \mathbb{S}(\boldsymbol{G}) \text{ where } S \subseteq S^+\}$.*

Proof. It follows directly from Proposition 11 and the recursive application of Theorem 13 (iii). □

In other words, a separator of a coarser DG belongs to the set of separators of the thinner DG or is a superset of one of its separators.

Next we prove that the cliques of a DG coarser than an MkDG are of equal size or higher than k and the cliques of size k are included in the MkDG.

Corollary 15. *Let $\boldsymbol{G}$ be an MkDG and $\boldsymbol{G}^+$ be a coarser DG, then*

i) the cliques of $\boldsymbol{G}^+$ are of equal size or higher than k, and

ii) the cliques of size k of $\boldsymbol{G}^+$ are included in the set of cliques of $\boldsymbol{G}_k$.

Proof. By Theorem 6, Proposition 11 and Corollary 14 we know that all of the separators of $\boldsymbol{G}^+$ are of size equal to or higher than $k-1$. Thus, by Theorem 4 the addition of a candidate edge can only generate cliques of size higher than k, which proves claim i). Furthermore, if no clique of size k can be created then all the cliques of size k must be contained in the set of cliques of $\boldsymbol{G}$, which proves claim ii). □

Now we focus on describing how to solve Problem 2. To do this we have to characterize the edges whose addition creates cliques of size $k+1$. By Theorem 4 these edges are determined by the candidate edges due to separators of size $k-1$ (see Section 3). Thus, in order to consider all the M$(k+1)$DGs coarser than a given MkDG, $\boldsymbol{G}_k$, we need to identify all the separators of size $k-1$ of any $(k+1)$DG coarser than $\boldsymbol{G}_k$.

Corollary 16. *Let $\boldsymbol{G}$ be an MkDG and $\boldsymbol{G}^+$ be a coarser DG, then all the separators of size $k-1$ of $\boldsymbol{G}^+$ are contained in the set of separators of $\boldsymbol{G}$.*

Proof. It follows directly from Theorem 6 and Corollary 14. □

The next result characterizes each candidate edge of an MkDG or any coarser DG that creates cliques of size $k+1$.

Corollary 17. *Let $\boldsymbol{G}$ be an MkDG and $\boldsymbol{G}^+$ be a DG coarser than (or equal to) $\boldsymbol{G}$. Then, $\mathbb{E}_{k+1}(\boldsymbol{G}^+) = \cup_{S \in \mathbb{S}(\boldsymbol{G})} \mathbb{E}_{\boldsymbol{G}^+}(S)$.*

Proof. It is a direct consequence of Theorem 4, Definition 7, Corollary 16 and the fact that $\mathbb{E}_{k+1}(\boldsymbol{G}^+)$ consists of the candidate edges that create cliques of size $k+1$. □

The previous result tells us that all the candidate edges $\mathbb{E}_{k+1}(\boldsymbol{G}^+)$ can be determined efficiently by a local inspection of the mantles of the separators of the MkDG $\boldsymbol{G}$.

Next we characterize the mantles of an MkDG, a coarser $(k+1)$DG and a coarser M$(k+1)$DG. This characterization justifies the use of the separator-based decomposition induced by the set of separators of an MkDG for approximating Problem 2.

Proposition 18. *Let $\boldsymbol{G}_k$ be an MkDG, $\boldsymbol{G}^+$ be a $(k+1)$DG coarser than $\boldsymbol{G}_k$ and $\boldsymbol{G}_{k+1}$ be a coarser M$(k+1)$DG. Then*

i) the mantle of any $S \in \mathbb{S}(\boldsymbol{G}_k)$ in $\boldsymbol{G}_k$, $\boldsymbol{G}_k[\mathbb{N}(S)]$, is the empty graph,
ii) the mantle of any $S \in \mathbb{S}(\boldsymbol{G}_k)$ in $\boldsymbol{G}^+$, $\boldsymbol{G}^+[\mathbb{N}(S)]$, is a forest, and
iii) the mantle of any $S \in \mathbb{S}(\boldsymbol{G}_k)$ in $\boldsymbol{G}_{k+1}$, $\boldsymbol{G}_{k+1}[\mathbb{N}(S)]$, is a tree.

Proof. We prove i), ii) and iii) by contradiction.

i) Assume, the contrary, that there is an edge $\{u, v\}$ in the mantle of a separator S of $\boldsymbol{G}_k$. Since we know that all the separators are of size $k-1$, $\boldsymbol{G}_k$ has the clique $\{u, v\} \cup S$ of size $k+1$. However, this contradicts the fact that all the cliques of an MkDG are of size k. Therefore, all the mantles are the empty graph.

ii) Assume, the contrary, that there exists a set $S \in \mathbb{S}(\boldsymbol{G}_k)$ of size $k-1$ with a clique C of size greater than 2 in its mantle. Then $\boldsymbol{G}^+$ has the clique $C \cup S$ of size greater than $k+2$. However, we know that $\boldsymbol{G}^+$ is a $(k+1)$DG and we know that the maximum clique size is $k+1$, but this contradicts the fact that C is of size greater than 2. Thus, the mantle of any set $S \in \mathbb{S}(\boldsymbol{G}_k)$ in $\boldsymbol{G}^+$ has cliques up to size 2. By Corollary 2.8 in Lauritzen (1996) we know that any mantle in a decomposable graph is also a decomposable graph and, therefore, the mantle is a forest.

iii) By Definition 5, M$(k+1)$DGs are special cases of $(k+1)$DGs for which the addition of a candidate edge produces a clique of a size higher than $k+1$. Thus by (ii) we know that for any $S \in \mathbb{S}(\boldsymbol{G}_k)$ we have that the mantle of S_i in $\boldsymbol{G}_{k+1}$ is a forest.

We prove that the forest is a tree by contradiction. Let us assume that the mantle of S is not a tree. Then at least the mantle of S has two distinct connected components. We can take any vertex u from one connected component and any vertex v from another connected component. Since u and v are separated by S and they belong to its common neighborhood, the edge $\{u, v\}$ is a candidate edge by Theorem 4. And, by Theorem 13 (i), the addition of $\{u, v\}$ to $\boldsymbol{G}_{k+1}$ creates a clique of size $k+1$ which contradicts Definition 5. Therefore, all the mantles are trees. □

It should be noted that the graph induced by a separator S and its common neighbors $\mathbb{N}(S)$, i) is an MkDG when the mantle of S is the empty graph, ii) is a $(k+1)$DG when the mantle is a forest, and iii) is an M$(k+1)$DG when the mantle is a tree.

Next, we show when and how the addition of an edge to the mantle of a separator could only affect the mantles of adjacent separators. This result is used by PFT in order to determine the interactions between the separators due to edge additions.

Proposition 19. *Let $\boldsymbol{G} = (V, E)$ be an MkDG or a $(k+1)$DG coarser than an MkDG. Let S and S' be two separators of size $k-1$ in $\boldsymbol{G}$, and let $\{u, v\}$ be a candidate edge due to S. The addition of $\{u, v\}$ to $\boldsymbol{G}$ modifies the mantle of S' if and only if u (or v) is contained in S' and $S \setminus S' = \{w\}$. If $u \in S'$ then the vertex v and the edge $\{v, w\}$ are added to the mantle of S.*

Proof. Let $\boldsymbol{G}^+$ be the graph obtained by adding $\{u, v\}$ to $\boldsymbol{G}$. By Theorem 4, $\{u, v\} \subseteq \mathbb{N}_{\boldsymbol{G}}(S)$ and the clique $C = S \cup \{u, v\}$ of size $k+1$ is created in $\boldsymbol{G}^+$. By Theorem 13 C is the unique clique created by the addition of $\{u, v\}$ and no more separators of size $k-1$ are created. Therefore, we concentrate in the modifications that cause C.

We start by proving $A \Rightarrow B$. Let us assume that $u \in S'$ and $S \setminus S' = \{w\}$:

- Since $|S| = |S'|$ and $u \notin S$, we have that $S' \cup \{w\} = S \cup \{u\}$. Then $w \in \mathbb{N}_{\boldsymbol{G}}(S')$ and thus w belongs to the mantle of S' in $\boldsymbol{G}'$.

- Since $S' \cup \{w\} = S \cup \{u\}$ we have that $C \setminus S' = \{v, w\}$. Then $\{v, w\} \subseteq \mathbb{N}_{\boldsymbol{G}^+}(S')$ and, thus, the vertices v and the edge $\{v, w\}$ is added to the mantle of S' in $\boldsymbol{G}^+$.

We conclude by proving $\neg A \nRightarrow \neg B$:

- Let us assume that u is not contained in S. Then $S' \not\subset S \cup \{u\}$ because S' and S differ at least in one element, $|S| = |S'| = k - 1$ and $u \notin S'$. Thus, u is not part of the mantle of S in $\boldsymbol{G}$ nor in $\boldsymbol{G}^+$. The same argument hold if we assume that v is not contained int S. Therefore, the mantle of S' is not modified by the addition of $\{u, v\}$.

- Let us assume that $S \setminus S' = R$ with $|R| > 1$. Note that $\{u, v\} \not\subseteq S'$ because S' is a complete set in $\boldsymbol{G}$. Then $S' \not\subseteq S \cup u$ because S and S' differs at least in two elements and $|S| = |S'| = k - 1$. Thus, u is not part from the mantle of S in $\boldsymbol{G}$ nor in $\boldsymbol{G}^+$. The same argument hold for v. Therefore, the mantle of S' is not modified by the addition of $\{u, v\}$.

□

It should be noted that the addition of the edge $\{u, v\}$ to the mantle of S can only produce modifications to the mantle of an adjacent separator S' by adding the new vertex u and the edge $\{u, w\}$, where $\{w\} = S \setminus S'$. Therefore the mantle of S' is a forest with the same number of connected components than in $\boldsymbol{G}$. Clearly, the addition of an edge to a mantle of a separator increases or keeps equal the candidate edges due to adjacent separators.

The following results guarantee that the sets of candidate edges due to different separators are disjoint during the execution of the (parallel and sequential) growing step of the family of fractal tree algorithms.

Proposition 20. *Let $\boldsymbol{G}$ be a DG and let $\boldsymbol{G}^+$ be an equal or coarser DG. Let S and R be two separators of $\boldsymbol{G}$, where $S \not\subseteq R$. Then $\mathbb{E}_{\boldsymbol{G}}(S) \cap \mathbb{E}_{\boldsymbol{G}^+}(R) = \emptyset$.*

Proof. We prove this result by contradiction. Let us assume that $\{u, v\} \in \mathbb{E}_{\boldsymbol{G}}(S) \cap \mathbb{E}_{\boldsymbol{G}^+}(R)$ exists. If $S \not\subseteq R$, then s exists, where $s \in S \setminus R$. By Definition 7, u and v are separated by S in $\boldsymbol{G}$ and by R in $\boldsymbol{G}^+$. Besides, by the same definition, u and v belong to the mantle of S in $\boldsymbol{G}$ and, then $\boldsymbol{G}$ includes the edges $\{u, s\}$ and $\{s, v\}$. Thus, $\boldsymbol{G}$ has the path u, s, v. Since $\boldsymbol{G}^+$ is coarser than $\boldsymbol{G}$, we have that R does not separate u and v in $\boldsymbol{G}^+$. Therefore, by Definition 7, $\{u, v\} \notin \mathbb{E}_{\boldsymbol{G}^+}(R)$, which contradicts the initial assumption. □

Next, we prove that the number of mutual information quantities computed by the growing steps of the FT algorithm are $\binom{n-k+1}{2}$, in the worst case. Since the computational complexity of the fractal tree family of algorithms is dominated by the computation of the mutual information quantities, this result can be used in order to determine the computational complexity of the fractal tree family of algorithms in the worst case (see Section 4.5).

Corollary 21. *Given an MkDG with n vertices, the growing step of the fractal tree family of algorithms requires to compute $\binom{n-k+1}{2}$, in the worst case. The worst case corresponds to an MkDG with a single separator.*

Proof. The amount of mutual information quantities that a growing step computes is the sum of the number of candidate edges due to the different separators considered by the procedure. The growing steps only consider the separators of the MkDG, which are of size $k - 1$. Every pair of distinct separators S and R of the MkDG satisfy that $R \not\subseteq S$ and $S \not\subseteq R$. Therefore,

by Proposition 20, we know that the sets of candidate edges due to two different separators are disjoint for the parallel and sequential growing steps. In other words, a candidate edge can be considered only due to a single separator.

By Theorem 6, an MkDG has $n-k+1$ cliques of size k and, by Theorem 4, we know that this set of cliques has been obtained from a thinner M$(k-1)$DG by adding $n-k+1$ (candidate) edges. Therefore, an MkDG has $\sum_{i=2}^{k} n-i+1$ edges out of the $\binom{n}{2}$ possible edges. Thus, in the worst case, at most $\binom{n}{2} - \sum_{i=2}^{k} n-i+1 = \binom{n-k+1}{2}$ mutual information quantities can be computed in a growing step. The worst case corresponds to the number of mutual information quantities computed for an MkDG with a single separator, which has $n-k+1$ vertices in its mantle. □

In the next results, we prove that the fractal tree family of algorithms produces a sequence of MiDG for $i = 2, .., k$. For this purpose we demonstrate that the i^{th} growing step produces an M$(i+1)$DG given an input MiDG. It should be noted that the growing procedures of FT create trees in all the mantles of all the separators of a given MiDG.

Proposition 22. *Let $\boldsymbol{G}$ be an MkDG or a $(k+1)$DG coarser than an MkDG with a separator S of size $k-1$. Let $\mathcal{E}$ be a (non-empty) set of edges that is added to $\boldsymbol{G}$ to form $\boldsymbol{G}^+$, where i) $\mathcal{E}$ is a subset of the candidate edges due to S in $\boldsymbol{G}$, and ii) the mantle of S in $\boldsymbol{G}^+$ is a tree. Then $\boldsymbol{G}^+$ is a $(k+1)$DG where $\mathbb{C}(\boldsymbol{G}^+) \setminus \mathbb{C}(\boldsymbol{G}) = \{S \cup \{u,v\} : \{u,v\} \in \mathcal{E}\}$.*

Proof. First, we prove that $\boldsymbol{G}^+$ is a DG and then we prove that it is a $(k+1)$DG.

By Proposition 18, the mantle of S is a forest in $\boldsymbol{G}$ with at least two connected components because it is a separator in $\boldsymbol{G}$. By Definition 7, $\boldsymbol{G}$ has at least one candidate edge due to S. Thus, there exists a (non-empty) set of candidate edges due to S in $\boldsymbol{G}$, $\mathcal{E}$, where its addition to $\boldsymbol{G}$ forms $\boldsymbol{G}^+$ and generates a mantle of S with tree structure.

Let $\{u,v\} \in \mathcal{E}$. If we add any subset of $\mathcal{E}$ to $\boldsymbol{G}$ that does not contain $\{u,v\}$ to form $\boldsymbol{G}^-$, then u and v will belong to different connected components in the mantle of S in $\boldsymbol{G}^-$. Then, by Definition 7, $\{u,v\} \in \mathbb{E}_{\boldsymbol{G}^-}(S)$ and, by Corollary 17, $\{u,v\} \in \mathbb{E}_k(\boldsymbol{G}^-)$. In other words, the addition of any subset of $\mathcal{E}$ to $\boldsymbol{G}$ that does not contain $\{u,v\}$ creates a DG for which $\{u,v\}$ is a candidate edge. Therefore, the addition of $\mathcal{E}$ to $\boldsymbol{G}$ can be understood as a sequential addition of candidate edges due to S and then, by Proposition 11, $\boldsymbol{G}$ is a DG.

Finally, by Theorem 13 (i), a candidate edge $\{u,v\} \in \mathcal{E}$ due to S creates the clique $S \cup \{u,v\}$ of size $k+1$. Therefore, $\boldsymbol{G}^+$ is a $(k+1)$DG with the set of cliques $\{S \cup \{u,v\} : \{u,v\} \in \mathcal{E}\}$ not contained in the set of cliques of $\boldsymbol{G}$. □

Theorem 23. *Let $\boldsymbol{G}^0$ be a MkDG with separators $S_1, ..., S_l$. Let $\boldsymbol{G}^l$ be a graph coarser than $\boldsymbol{G}^0$ that has been obtained by constructing a sequence of graphs $\boldsymbol{G}^0, \boldsymbol{G}^1, ..., \boldsymbol{G}^l$. $\boldsymbol{G}^i$ for $i = 1, ..., l$ has been obtained from $\boldsymbol{G}^{i-1}$ by adding a (non-empty) set of candidate edges $\mathcal{E}_i$ due to S_i, where its addition to $\boldsymbol{G}^{i-1}$ generates a mantle of S_i with tree structure. Then $\boldsymbol{G}^l$ is an M$(k+1)$DG.*

Proof. First, we prove by induction over j, for $j = 1, .., l$, that $\boldsymbol{G}^l$ is a $(k+1)$DG, for which the mantles of $S_1, ..., S_l$ are trees. Then we prove that, hence, $\boldsymbol{G}^l$ is an M$(k+1)$DG.

Let $j = 1$. $\boldsymbol{G}^1$ has been obtained from $\boldsymbol{G}^0$ by adding a set of candidate edges $\mathcal{E}_1$ due to S in $\boldsymbol{G}^0$, where the mantle for S_1 in $\boldsymbol{G}^1$ is a tree. By Proposition 22, $\boldsymbol{G}^1$ is a $(k+1)$DG. The addition of $\mathcal{E}_1$ to $\boldsymbol{G}^0$ can cause the modification of the mantles of other separators. However, by Proposition 18 (ii), the mantles of $S_2, ..., S_l$ in $\boldsymbol{G}^1$ are forests and they have the same number of connected components as in $\boldsymbol{G}^0$ –at least two.

Let $j = i$, we assume that $\boldsymbol{G}^i$ is a $(k+1)$DG, the mantles of $S_1, ..., S_i$ are trees in $\boldsymbol{G}^i$ and the mantles of $S_{i+1}, ..., S_l$ in $\boldsymbol{G}^i$ are forests with at least two connected components. Thus, there exists a (non-empty) subset of candidate edges $\mathcal{E}_{i+1}$ due to S_{i+1} in $\boldsymbol{G}^i$, whose addition to $\boldsymbol{G}^i$ creates $\boldsymbol{G}^{i+1}$, and the mantle of S_{i+1} in $\boldsymbol{G}^{i+1}$ is a tree. By Proposition 22, $\boldsymbol{G}^{i+1}$ is a $(k+1)$DG. Again, by Proposition 18 (ii), the mantles of $S_1, ..., S_i$ are trees in $\boldsymbol{G}^{i+1}$. Besides, by the same proposition the mantles of $S_{i+2}, ..., S_l$ are forests in $\boldsymbol{G}^{i+1}$ with the same number of connected components as in $\boldsymbol{G}^i$ –at least two.

By the principle of induction, $\boldsymbol{G}^l$ is a $(k+1)$DG for which the mantles of $S_1, ..., S_l$ are trees.

Next, we prove that all the cliques of $\boldsymbol{G}^l$ are of size $k+1$ and that the addition of a candidate edge to $\boldsymbol{G}^l$ creates a clique of a size higher than $k+1$, which by Definition 5 proves that $\boldsymbol{G}^l$ is an M$(k+1)$DG. We know that mantle of S_i for $i = 1, ..., l$ in $\boldsymbol{G}^l$ is a tree. A tree has a single connected component and, thus, all the vertices in the mantle of S_i are included in at least one edge from the mantle. Besides, by the definition of chain of cliques, we know that for any clique C in $\boldsymbol{G}^0$ we have (at least) one separator $S \in \{S_1, .., S_l\}$ where $C = S \cup \{u\}$. Then, all the cliques of $\boldsymbol{G}^0$ are included in at least one clique of size $k+1$ of $\boldsymbol{G}^l$. And therefore, all the cliques of $\boldsymbol{G}^l$ are of size $k+1$.

In addition, since the mantle of S_i for $i = 1, ..., l$ in $\boldsymbol{G}^l$ is a tree, by Definition 7, the set of candidate edges due to S_i in $\boldsymbol{G}^l$ is empty. Then, by Theorem 6 (ii), Corollary 16 and Corollary 17, the set of candidate edges for $\boldsymbol{G}^l$ that create cliques of size $k+1$ is empty. Thus, the addition of a candidate edge to $\boldsymbol{G}^l$ creates a clique of a size greater than $k+1$. Therefore, by Definition 5, $\boldsymbol{G}^l$ is an M$(k+1)$DG. □

This result closely resembles the sequential growing step of SFT but it can be directly applied to the parallel growing step of PFT. PFT does not take into account the interactions among the separators. However, by Proposition 19 the interactions among the mantles of the separators can only increase the set of candidate edges due to a separator. Thus, the parallel growing step can also produce a sequence of structures by constructing trees at each separator sequentially.

Finally, we prove that pruning a leaf vertex[12] from a given MkDG and grafting it into a separator produces an MkDG.

Proposition 24. *Let $\boldsymbol{G}$ be an MkDG with a leaf u that belongs to the clique $S \cup \{u\}$. If we prune u from S and then we graft u into a separator S' of $\boldsymbol{G}$ (possibly $S' = S$), we produce an MkDG with an entropy reduced by $\hat{I}(X_u; \boldsymbol{X}_{S'}) - \hat{I}(X_u; \boldsymbol{X}_S)$.*

Proof. We start by proving that pruning a leaf vertex from its separator and grafting it in a separator produces an MkDG. Let u be a leaf vertex of the clique C which belongs to the separator S, then $C = S \cup \{u\}$. Due to Corollary 2.8 in (Lauritzen, 1996), the subgraph induced by $V \setminus \{v\}$, $\boldsymbol{G}^-$, is a decomposable graph. Moreover, all the cliques of $\boldsymbol{G}^-$ are of size $k-1$ and all of its separators of size $k-1$. Then, by Definition 5, it is an MkDG. If we choose any possible separator S' in the graph $\boldsymbol{G}^-$ and we create the clique $S' \cup \{u\}$ to form $\boldsymbol{G}'$, then by Lemma 2.19 in (Lauritzen, 1996) $\boldsymbol{G}'$ is a decomposable graph. In addition, all its cliques are of size k and all of its separators are of size $k-1$. Therefore, by Definition 5, is an MkDG.

Next, we prove the reduction in the entropy produced by pruning and grafting u. The pruning of u from the separator S, reduces the entropy of the model in $\hat{H}(X_u|\boldsymbol{X}_S) =$
$\hat{H}(X_u) - \hat{I}(X_u; \boldsymbol{X}_S)$, where $\hat{H}$ and $\hat{I}$ denote the empirical entropy and mutual information

[12] i.e., a vertex that belongs to a single clique

respectively. To graft u in S' increases the entropy of the graph in $\hat{H}(X_u|\boldsymbol{X}_{S'}) = H(X_u) - \hat{I}(X_u; \boldsymbol{X}_{S'})$. Therefore, after pruning u from S and grafting into S', the entropy of the structure is reduced on $\hat{I}(X_u; \boldsymbol{X}'_S) - \hat{I}(X_u; \boldsymbol{X}_S)$. □

In consequence, P&G (see Algorithm 3) produces an MkDG structure with an equal or lower entropy or, in other words, an equal or higher likelihood.

Appendix B: Artificial domains

Algorithm 5 shows the pseudocode of theprocedure used to generate the artificial domains with MkDG structure used in the experiments of Section 6.2.

Algorithm 5. *(MkDG random generator)*
Input: *n and k.*
Output: *An MkDG.*
Pseudocode:

1. $\mathcal{C} = \{\{1, ..., k\}\}$
2. `For` $i = k + 1, ..., n$
3. `-Take a clique` C `at random from` $\mathcal{C}$
4. `-Take a subset` S `of size` $k - 1$ `from` C `at random.`
5. `-`$\mathcal{C} = \mathcal{C} \cup \{S \cup \{i\}\}$
6. `return` (V, E) `where` $V = \{1, ..., n\}$ `and` $E = \{\{u, v\} : \exists C \in \mathcal{C},$ `where` $\{u, v\} \subseteq C\}$

Algorithm 5 shows the pseudocode of theprocedure used to generate the artificial domains with sDG structure used in the experiments of Section 6.3.

Algorithm 6. *(sDG random generator)*
Input: *n and s.*
Output: *An sDG.*
Pseudocode:

1. $C = \{1, ..., rand(2, ..., s)\}$, $remain = \{1, ..., n\} \setminus C$ `and` $\mathcal{C} = \{C\}$
2. `While` $remain \neq \emptyset$
3. `-Take a clique` C `at random from` $\mathcal{C}$
4. `-Take a subset` S `of size` $rand(1, ..., |C| - 1)$ `from` C `at random`
5. `-Take a subset` R `size` $rand(1, ..., s - |S|)$ `from` $remain$ `at random`
6. `-`$\mathcal{C} = \mathcal{C} \cup \{S \cup R\}$ `and` $remain = remain \setminus R$
7. `return` (V, E) `where` $V = \{1, ..., n\}$ `and` $E = \{\{u, v\} : \exists C \in \mathcal{C},$ `where` $\{u, v\} \subseteq C\}$

References

Bach, F. R., Jordan, M. I., 2001. Thin junction trees. In: Advances in Neural Information Processing Systems (NIPS). pp. 569–576.

Chechetka, A., Guestrin, C., 2008. Efficient principled learning of thin junction trees. In: Advances in Neural Information Processing Systems (NIPS). pp. 273–280.

Chow, C., Liu, C., 1968. Approximating discrete probability distributions with dependence trees. IEEE Transactions on Information Theory 14, 462–467.

Chow, C., Wagner, T., 1971. Consistency of an estimate of tree-dependent probability distribution. IEEE Transactions on Information Theory 19 (3), 369–371.

Corander, J., Janhunen, T., Rintanen, J., Nyman, H. L., Pensar, J., 2013. Learning chordal markov networks by constraint satisfaction. In: Advances in Neural Information Processing Systems (NIPS). pp. 1349–1357.

Desphande, A., Garofalakis, M., Jordan, M. I., 2001. Efficient stepwise selection in decomposable models. In: Uncertainty and Artificial Intelligence. pp. 128–135.

Ding, G., Lax, R. F., Chen, J., Chen, P. P., Marx, B. D., 2007. Comparison of Greedy Strategies for Learning Markov Networks of Treewidth k. In MLMTA (p. 294).

Eisner, J., 1997. State-of-the-art algorithms for minimum spanning trees. Research report, University of Pennsylvania.

Højsgaard, S., Edwards, D., Lauritzen, S., 2012. Graphical Models with R. Springer.

Jordan, M., et al., 2013. Frontiers in Massive Data Analysis. The National Academies Press.

Kangas, K., Miinimäki, T., Koivisto, M., 2014. Learning chordal markov networks by dynamic programming. In: Advances in Neural Information Processing Systems (NIPS). pp. 2357–2365.

Karger, D., Srebro, N., 2001. Learning markov networks: Maximum bounded tree-width graphs. In: Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, pp. 392–401.

Koller, D., Friedman, N., 2009. Probabilistic Graphical Models. Principles and Techniques. The MIT Press, Cambridge, Massachusetts.

Kovács, E., Szántai, T., 2010. On the approximation of discrete multivariate probability distribution using the new concept of t-cherry junction tree. In: Coping with Uncertainty, Lecture Notes in Economics and Mathematical Systems. Vol. 633. pp. 39–56.

Lauritzen, S. L., 1996. Graphical Models. Oxford University Press, New York.

Lauritzen, S. L., Spped, T. P., Vijayan, K., 1984. Decomposable graphs and hypergraphs. Journal of Australian Mathematical Society A 36, 12–29.

Madsen, A., Jensen, F., Salmerón, A., Karlsen, M., Langseth, H., Nielsen, T., 2014. A new method for vertical parallelisation of TAN learning based on balanced incomplete block designs. In: 7th European Workshop on Probabilistic Graphical Models. Vol. 8754 of Lecture Notes in Computer Sciences. pp. 302–3017.

Malvestuto, F. M., 1991. Approximating discrete probability distributions with decomposable models. IEEE Transactions on Systems, Man and Cybernetics 21 (5), 1287–1294.

Malvestuto, F. M., 2012. A backward selection procedure for approximating a discrete probability distribution by decomposable models. Kybernetica 48 (5), 825–844.

Pearl, J., 1988. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Mateo, California.

Proulx, B., Zhang, J., 2014. Modeling social networks relationships via t-cherry junction trees. In: The 33rd IEEE Conference on Computer Communications. INFOCOM. pp. 2229–2237.

Rodríguez, J. D., Pérez, A., Lozano, J. A. (2013). A general framework for the statistical analysis of the sources of variance for classification error estimators. Pattern recognition, 46(3), 855-864.

Srebro, N., 2000. Maximum likelihood markov networks: An algorithmic approach. Master thesis, Massachuset Institute of Technology.

Srebro, N., 2003. Maximum likelihood bounded tree-width markov networks. Artificial Intelligence 143, 123–138.

Szántai, T., Kovács, E., 2011. Discovering a junction tree behind a markov network by a greedy algorithm. arXiv 1104.2762v3, 1–16.

Szántai, T., Kovács, E., 2012. Hypergraphs as a mean of discovering the dependence structure of a discrete multivariate probability distribution. Ann Oper Res 193, 71–90.