Technical Report

UNIVERSITY OF THE BASQUE COUNTRY
Department of Computer Science and Artificial Intelligence

Efficient learning of decomposable models with bounded clique size

Aritz Pérez, Iñaki Inza, Jose A. Lozano

May, 2014

San Sebastian, Spain
www.ehu.es/ccia-kzaa
hdl.handle.net/10810/4562

Efficient learning of decomposable models with a bounded clique size

Aritz Pérez aritz.perez@ehu.es
Iñaki Inza inaki.inza@ehu.es
Jose A. Lozano ja.lozano@ehu.es
Department of Computer Science and Artificial Intelligence
University of the Basque Country
San Sebastián, Spain

May 2, 2014

Abstract

The learning of probability distributions from data is a ubiquitous problem in the fields of Statistics and Artificial Intelligence. During the last decades several learning algorithms have been proposed to learn probability distributions based on decomposable models due to their advantageous theoretical properties. Some of these algorithms can be used to search for a maximum likelihood decomposable model with a given maximum clique size, k, which controls the complexity of the model. 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 a family of algorithms which approximates this problem with a computational complexity of $\mathcal{O}(k \cdot n^2 \log n)$ in the worst case, where n is the number of implied random variables.

The structures of the decomposable models that solve the maximum likelihood problem are called maximal k-order decomposable graphs. Our proposals, called fractal trees, 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 based on the particular features of this type of structures. Additionally, we propose a prune-and-graft procedure which transforms a maximal k-order decomposable graph into another one, increasing its likelihood. We have implemented two particular fractal tree algorithms called parallel fractal tree and sequential fractal tree. These algorithms can be considered a natural extension of Chow and Liu's algorithm, from $k = 2$ to arbitrary values of k. Both algorithms have been compared against other efficient approaches in artificial and real domains, and they have

shown a competitive behavior to deal with the maximum likelihood problem. Due to their low computational complexity they are especially recommended to deal with high dimensional domains.

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 approach 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 robust probabilistic models with a low computational complexity in order to efficiently perform probabilistic inference tasks. During the last decades, probabilistic graphical models have provided one of the most effective tools for the automatic learning of probabilistic models [15], being the family of decomposable models one of the most attractive due to their advantageous theoretical properties [10]. 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 criteria [9], among others. Our contributions consist of quantitative algorithms based on the maximization of the likelihood score.

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, i.e., the higher the likelihood, the lower the Kullback-Leibler divergence. It is known that the likelihood of a model tends to be higher as its complexity increases, because the fitting capability 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 goodness 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 phenomena. 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.

One of the most popular quantitative approaches based on the likelihood score for learning probability distributions with a low number of parameters is Chow and Liu's algorithm (**CL**, [3]). The algorithm can be implemented using Kruskal's algorithm for maximization, where the edge weights correspond to the empirical mutual information between the implied random variables. 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 proved that the algorithm is asymptotically consistent ([4]). Finally, the algorithm has a computational complexity of only $\mathcal{O}(n^2 \log n)$, where n is the number of implied random variables. Thus, due to its low computational complexity, the huge number of candidate models and its optimality, the algorithm is an excellent building block for designing novel search strategies focused on the maximization of the likelihood.

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 model in terms of conditional independences using a graphical criteria, and that they are the basis of the most popular inference algorithms over probability distributions [10]. During the last decades many quantitative approaches have been proposed in order to learn decomposable models. We would like to emphasize the seminal work of [13], 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 demonstrated that the structure that maximizes the likelihood, given a maximum clique size, is a maximum k-order decomposable graph (MkDG, see Definition 4). It proposes two greedy structural learning procedures of decomposable models, based on forward and backward searches, which consider single edge modifications. In [5] the authors present, based on the

results provided by [10], a formal characterization of the set of edges that can be added to a decomposable graph maintaining its decomposability, and design an algorithm for its identification with a computational complexity of $\mathcal{O}(n^2)$. The work of [16, 17] demonstrates that the learning of maximum likelihood decomposable models, with k greater than 2 and lower than $n-1$, is NP-hard. Thus, unless $P \equiv NP$, there is not a polynomial time algorithm for solving the problem and it has to be approached. Unfortunately, most of the algorithms proposed for learning decomposable models with a maximum clique size k [8, 1, 2] are exponential in k and, thus, they are unpractical for approaching high dimensional probability distributions, even for moderate values of k.

In this work, we propose a family of efficient algorithms, called fractal tree, for learning MkDGs (see Section 5). Our family of algorithms follow a divide-and-conquer strategy based on the particular structural features of MkDGs. The algorithms construct a sequence of maximal i-order decomposable graphs, for $i = 2, ..., k$, in $k-1$ steps. At each step, the input maximal i-order decomposable graph is divided into subgraphs related to the neighborhood of its separators. Then, the neighborhood of each separator is solved using a novel extension of CL, called generalized Chou and Liu (see Section 4). The partial solutions to each neighborhood are added to the input structure for generating the next maximal $(i+1)$-order decomposable graph. Additionally, we have developed a prune-and-graft operator for MkDGs that favors the mobility of the vertices across the structure. This procedure transforms an MkDG into another MkDG with a likelihood equal or higher. The prune-and-graft procedure can be directly plugged in the fractal tree algorithms and tends to improve the likelihood of the final resulting MkDG.

In this work, two variants of fractal tree are proposed, the parallel fractal tree and the sequential fractal tree. Parallel fractal tree solves all the neighborhoods of the separators in parallel, without considering the interactions among them, while sequential fractal tree solves them sequentially, taking into consideration their interactions. The implementation of sequential fractal tree includes the aforementioned prune-and-graft procedure. Both algorithms have a computational complexity of $\mathcal{O}(k \cdot n^2 \log n)$, in the worst case, which is k times the computational complexity of the CL algorithm ($\mathcal{O}(n^2 \log n)$). In the performed experimental comparison, the fractal tree algorithms have shown a competitive performance compared to other state of the art algorithms. The efficiency of the proposed algorithms and the good results achieved make them recommendable to deal with the maximum likelihood problem, especially for approaching high dimensional probability distributions.

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 Sec-

tion 3 we present the intuitions and the justification behind the fractal tree algorithms. Section 4 formally defines, what we call, the separator problem. Besides, we present a natural extension of the CL algorithm to solve the separator problem. This algorithm inherits the theoretical properties of the CL algorithm. In Section 5 we present a family of algorithms called fractal tree which are constructed using the extension of CL as the building block, following a divide-and-conquer strategy. Two particular implementations of the family of fractal tree algorithms are proposed: parallel fractal tree and sequential fractal tree. Section 6 presents a prune-and-graft procedure which transforms an MkDGs into another with equal or higher likelihood score. This procedure can be applied to any fractal tree algorithm for improving its behavior to deal with the maximum likelihood problem. Section 7 summarizes the experimental results obtained by the proposed algorithms in 33 real and 900 artificial domains. Finally, in Section 8 we present the conclusion of this work highlighting the major contributions. Besides, we indicate the principal future work lines regarding the fractal tree algorithms and their applications.

2 Background

We denote by $\boldsymbol{X} = (X_1, ..., X_n)$ a 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 $i = 1, ..., n$. The random variable X_i takes values in the finite space Ω_i and the n-dimensional random variable $\boldsymbol{X}$ takes values in the space $\Omega = \Omega_1 \times ... \times \Omega_n$. 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. $\hat{p} = \arg_q \max \prod_{i=1}^{N} q(\boldsymbol{x}^i)$.

2.1 Decomposable graphs and models

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

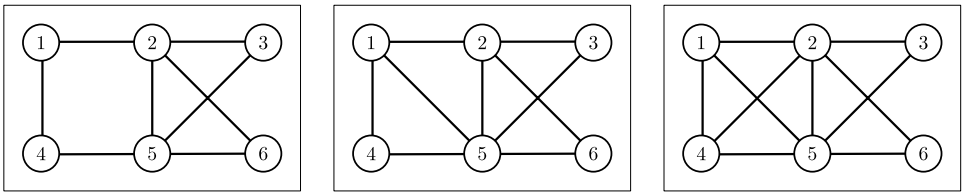
likelihood decomposable models with a bounded clique size.

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 any possible edge, i.e. $E = \{\{u, v\} : \{u, v\} \subset V\}$. 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 $E(V') = \{\{u, v\} : \{u, v\} \subset V' \wedge \{u, v\} \in E\}$. The examples introduced in this section are based on the graphs $\boldsymbol{G}^-$, $\boldsymbol{G}_3$ and $\boldsymbol{G}^+$ presented in Figure 1.

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, we say that $\boldsymbol{G}_3$ is coarser than $\boldsymbol{G}^-$ and thinner than $\boldsymbol{G}^+$.

The **neighborhood** of u in $\boldsymbol{G}$ is the set of vertices connected by an edge to u, $\{v \in V : \{u, v\} \in E\}$. It is denoted by $\mathbb{N}(u|\boldsymbol{G})$ or simply by $\mathbb{N}(u)$, when the graph is clear from the context. We define the neighborhood of a set of vertices S in $\boldsymbol{G}$ as the set of vertices connected by edges to all the vertices in S, $\bigcap_{u \in S} \mathbb{N}(u)$. Note that this definition of neighborhood is the intersection of the neighborhood of all the vertices instead of the union. The neighborhood of S is denoted by $\mathbb{N}(S|\boldsymbol{G})$ or simply by $\mathbb{N}(S)$, when the graph is clear from the context. For example, $\mathbb{N}(\{2, 5\}|\boldsymbol{G}^-) = \{3, 6\}$, $\mathbb{N}(\{2, 5\}|\boldsymbol{G}_3) = \{1, 3, 6\}$ and $\mathbb{N}(\{2, 5\}|\boldsymbol{G}^+) = \{1, 3, 4, 6\}$ (see Figure 1).



(a) Non-decomposable graph, $\boldsymbol{G}^-$.

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

(c) 4-order decomposable graph, $\boldsymbol{G}^+$.

Figure 1: Examples of undirected graphs

A sequence of vertices of V, $u_1, ..., u_l$, is a path of length $l-1$ when $\{u_i, u_{i+1}\}$ is included in E for $i = 1, ..., l-1$. A cycle is a path $u_1, ..., u_l$ for which $u_1 = u_l$. A chord of a cycle is an edge among two vertices which are not adjacent in the cycle. For example, the cycle $1, 2, 5, 4, 1$ in $\boldsymbol{G}_3$ has the chord $\{1, 5\}$. Two vertices u and v are said to be connected if there is a path from u to v. A connected component of

a graph is a subgraph induced by $V' \subseteq V$ in which any two vertices are connected to each other, and are not connected to vertices in $V \setminus V'$ in the original graph.

A **tree** is a graph with $n-1$ edges without cycles. For n vertices there are n^{n-2} different trees. A **forest** is a graph where each of its connected components is a tree.

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. From here on we will call the minimal separators, separators, for the sake of brevity. For example, the set $\{2, 4\}$ is the (minimal) separator for 1 and 5 in $\boldsymbol{G}^-$ but it does not separate them in $\boldsymbol{G}_3$.

Any $C \subseteq V$ is called a **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. The set of cliques associated to an undirected graph $\boldsymbol{G}$ is denoted as $\mathbb{C}(\boldsymbol{G})$. For example, $\mathbb{C}(\boldsymbol{G}^-) = \{\{1,2\},\{1,4\},\{4,5\},\{2,3,5\},\{2,5,6\}\}$, $\mathbb{C}(\boldsymbol{G}_3) = \{\{1,2,5\},\{1,4,5\},\{2,3,5\},\{2,5,6\}\}$ and $\mathbb{C}(\boldsymbol{G}^+) = \{\{1,2,4,5\},\{2,3,5\},\{2,5,6\}\}$. A subset of $\mathbb{C}(\boldsymbol{G})$ is denoted as $\mathcal{C}$. For sake of brevity $\bigcup_{C\in\mathcal{C}} C$ and $\bigcap_{C\in\mathcal{C}} C$ will be denoted as $\bigcup \mathcal{C}$ and $\bigcap \mathcal{C}$, respectively.

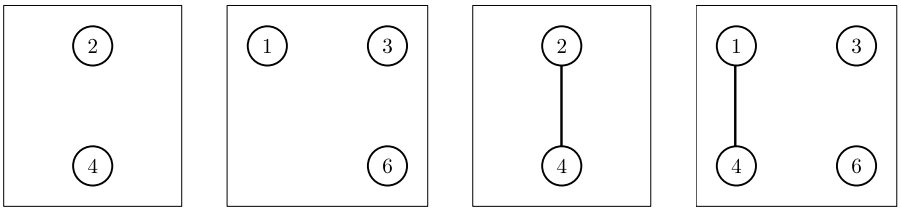
We say that an undirected graph is a **decomposable graph** (**DG**) if for any cycle of length greater than 3 there exists a chord. For example, $\boldsymbol{G}^-$ is not a DG because the cycle $1, 2, 5, 4, 1$ does not contain any chord. On the contrary, the cycle $1, 2, 5, 4, 1$ in $\boldsymbol{G}_3$ contains the chord $\{1, 5\}$. DGs are the graphical structures of decomposable models.

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 proved that a graph $\boldsymbol{G}$ is a DG if and only if it has a chain of cliques. For example, for $\boldsymbol{G}_3$ there exists the chain of cliques $C_1 = \{1,2,5\}, C_2 = \{1,4,5\}, C_3 = \{2,3,5\}, C_4 = \{2,6,5\}$ where $S_2 = \{1,5\} \subset C_1$, $S_3 = \{2,5\} \subset C_1$ and $S_4 = \{2,5\} \subset C_3$ or C_2. For $\boldsymbol{G}^-$ there is not a chain of cliques because the separator $\{2, 4\}$ (or $\{1, 5\}$) is not contained in any clique.

In DGs, the sets S_i for $i = 2, ..., m$ are the (minimal) separators. In this type of graphs any separator induces a complete subgraph and it is contained in at least two cliques. We denote the set of separators associated to the DG $\boldsymbol{G}$ as $\mathbb{S}(\boldsymbol{G})$, which consists of all the separators of the sequence $S_2, ..., S_m$ without repetition. The set of separators is unique for a DG.

Definition 2. *Let $\boldsymbol{G} = (V, E)$ be a DG, and let $S \subset V$ be a subset of vertices. The subgraph induced by the neighborhood of S, $\boldsymbol{G}(\mathbb{N}(S))$, is called the **mantle** of S for $\boldsymbol{G}$.*

The concept of mantle plays an important role in the divide-and-conquer strategy used by the fractal tree family of algorithms. Figure 2 illustrates the concept of mantle by means of four examples.



(a) $\boldsymbol{G}_3(\mathbb{N}(\{1,5\}))$. (b) $\boldsymbol{G}_3(\mathbb{N}(\{2,5\}))$. (c) $\boldsymbol{G}^+(\mathbb{N}(\{1,5\}))$. (d) $\boldsymbol{G}^+(\mathbb{N}(\{2,5\}))$.

Figure 2: The mantles of $\{1,5\}$ and $\{2,5\}$ in the graphs $\boldsymbol{G}_3$ (Figure 1b) and $\boldsymbol{G}^+$ (Figure 1c).

Next, we introduce a theoretical characterization, adapted from [5], of the set of the edges that can be added to a DG maintaining its decomposability. These edges are called **candidate edges**.

Theorem 1. *[5] 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 $\{u, v\} \subset \mathbb{N}(S)$.*

The set of candidate edges for $\boldsymbol{G}$ which does not create cliques of size greater than k is denoted as $\mathbb{E}_k(\boldsymbol{G})$.

Observation 1. *In order to determine $\mathbb{E}_k(\boldsymbol{G})$, we require to inspect the neighborhood of the separators of size lower than $k-1$.*

This observation we can used in order to design efficient procedures for determining $\mathbb{E}_k(\boldsymbol{G})$ (see Section 3).

Next, we introduce two types of decomposable graphs which control explicitly their maximum clique size.

Definition 3. *A k-**order decomposable graph** (k**DG**) is a decomposable graph for which the maximum clique size is k.*

For example, $\boldsymbol{G}_3$ is a 3DGs, and $\boldsymbol{G}^+$ is a 4DG.

Definition 4. *A **maximal** k-**order decomposable graph** (**M**k**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 $k+1$.*

MkDGs are also known in the literature as $(k-1)$-hypertrees [17]. The empty graph is an M1DG and a tree is an M2DG. We are interested in MkDGs because the solutions to Problem 1 are based on this family of structures (see Theorem 2 in Section 2.2). An MkDG $\boldsymbol{G}_k$ has the following interesting structural properties, among others [13]:

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

We can observe these properties in $\boldsymbol{G}_3$ which is an M3DG: $\mathbb{C}(\boldsymbol{G}_3) = \{\{1,2,5\}, \{1,4,5\}, \{2,3,5\}, \{2,5,6\}\}$ and $\mathbb{S}(\boldsymbol{G}_3) = \{\{1,5\}, \{2,5\}\}$, where $|\mathbb{C}(\boldsymbol{G}_3)| = 6 - 3 + 1 = 4$

MkDGs are maximal in the sense that no more cliques of size k can be constructed by adding candidate edges and they can also be characterized in terms of this property:

Corollary 1. *Let $\boldsymbol{G} = (V, E)$ be a DG. $\boldsymbol{G}$ is an MkDG if and only if $\mathbb{E}_k(\boldsymbol{G}) = \emptyset$*

Proof. Clearly, if $\mathbb{E}_k(\boldsymbol{G}) \neq \emptyset$ we can add any edge from $\mathbb{E}_k(\boldsymbol{G})$ without creating a clique of size greater than k which contradicts Definition 4. □

Next, we present 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** [11] 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, $\mathcal{P}_{\boldsymbol{G}} = \{p(\boldsymbol{X}_C) : C \in \mathbb{C}(\boldsymbol{G})\} \cup \{p(\boldsymbol{X}_S) : S \in \mathbb{S}(\boldsymbol{G})\}$. 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 $C, C' \in \mathbb{C}(\boldsymbol{G})$ where $C \cap C' = S$. The decomposable model $\boldsymbol{M}$ represents the following factorization of the joint probability distribution $p(\boldsymbol{x})$: $p_{\boldsymbol{M}}(\boldsymbol{x}) = \prod_{C \in \mathbb{C}(\boldsymbol{G})} p(\boldsymbol{x}_C) / \prod_{S \in \mathbb{S}(\boldsymbol{G})} p(\boldsymbol{x}_S)^{d_S}$, where d_S is the number of cliques that contain S minus one. We call the decomposable models with kDG and MkDG structures, k-order decomposable models and maximum 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 [10]. Besides, the set of maximum likelihood probability distributions can be computed in a closed form for decomposable models [10], 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 refer the model given by $\boldsymbol{G}$ and the set of maximum likelihood probabilities as the structure $\boldsymbol{G}$. Thus, the structure represents the following factorized probability distribution:

$$p_{\boldsymbol{G}}(\boldsymbol{x}) = \frac{\prod_{C \in \mathbb{C}(\boldsymbol{G})} \hat{p}(\boldsymbol{x}_C)}{\prod_{S \in \mathbb{S}(\boldsymbol{G})} \hat{p}(\boldsymbol{x}_S)^{d_S}} \tag{1}$$

The use of the structure instead of the model in the notation will emphasize that once the use of the empirical probability distributions is decided upon, the likelihood is a function of the network structure, $\boldsymbol{G}$, only. For example, $p_{\boldsymbol{G}_3}(\boldsymbol{x}) = \hat{p}(\boldsymbol{x}_{\{1,2,5\}})\hat{p}(\boldsymbol{x}_{\{1,4,5\}})\hat{p}(\boldsymbol{x}_{\{2,3,5\}})\hat{p}(\boldsymbol{x}_{\{2,5,6\}})/\hat{p}(\boldsymbol{x}_{\{1,5\}})\ \hat{p}(\boldsymbol{x}_{\{2,5\}})^2$ and $p_{\boldsymbol{G}^+}(\boldsymbol{x}) = \hat{p}(\boldsymbol{x}_{\{1,2,4,5\}})\hat{p}(\boldsymbol{x}_{\{2,3,5\}})\hat{p}(\boldsymbol{x}_{\{2,5,6\}})/\hat{p}(\boldsymbol{x}_{\{2,5\}})^2$

2.2 Learning maximum likelihood k-order decomposable models problem

This section formally defines the problem of learning maximum likelihood k-order decomposable models, and introduces a set of previous theoretical results which motivate the fractal tree algorithms presented in Section 5.

Problem 1. *Let $\mathcal{D} = \{\boldsymbol{x}^1, ..., \boldsymbol{x}^N\}$ be a 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:*

$$\boldsymbol{G}^* = \arg_{\boldsymbol{G} \in \mathcal{G}_k} \max \prod_{\boldsymbol{x} \in \mathcal{D}} p_{\boldsymbol{G}}(\boldsymbol{x}) \tag{2}$$

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

Let $\boldsymbol{G}$ and $\boldsymbol{G}^+$ be two kDGs, where $\boldsymbol{G} \prec \boldsymbol{G}^+$. It can be proved that $\boldsymbol{G}^+$ has a likelihood equal or higher than $\boldsymbol{G}$ [13]. Following this intuition, it is easy to prove that the solution to the problem of learning a maximum likelihood kDG is an MkDGs.

Theorem 2. *[13] The solution to Problem 1 is a maximal k-order decomposable model.*

Consequently, our approaches restrict the search of maximum likelihood kDG to the class of MkDGs.

The learning of a maximum likelihood MkDG for $k = 2$ can be solved polynomially using the CL algorithm [3]. However, this is not true for any value of k unless $P \equiv NP$.

Theorem 3. *[16, 17] Problem 1 is NP-hard for $k > 2$.*

In this work we propose tractable suboptimal algorithms in order to deal with Problem 1, where their computational complexity is $\mathcal{O}(k \cdot n^2 \log n)$, in the worst case.

The next theoretical result indicates the equivalence between Problem 1 and finding a minimum entropy MkDG.

Theorem 4. *[13] Problem 1 is equivalent to finding the k-order decomposable graph which minimizes the entropy:*

$$\arg_{\boldsymbol{G} \in \mathcal{G}_k} \max \prod_{\boldsymbol{x} \in \mathcal{D}} p_{\boldsymbol{G}}(\boldsymbol{x}) \equiv \arg_{\boldsymbol{G} \in \mathcal{G}_k} \min H_{\boldsymbol{G}}(\boldsymbol{X}) \tag{3}$$

where $H_{\boldsymbol{G}}(\boldsymbol{X})$ represents the entropy of $\boldsymbol{X}$ distributed according to $p_{\boldsymbol{G}}$.

From here on we will deal with Problem 1 through the minimum entropy formulation. 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 this problem, 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 3), where $\boldsymbol{G}_k$ could be a good approximation to Problem 1. Unfortunately, Problem 2 still remains an NP-hard for $k > 2$.

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

Proof. The proof is equivalent to the proofs provided in [16], Theorem 4.1, and [17], Corollary 3, for Problem 1. Clearly, we can reduce 3SAT to Problem 2, by constructing an MkDG coarser than the core structure given in Theorem 4.1 of [16]. □

The fractal tree algorithms (see Section 5) approach Problem 2 using a divide-and-conquer strategy for learning an M$(i+1)$DG coarser than a given MiDG. The MiDG is divided into separators of size $k-1$. Then, the problem associated to each of these separators is solved by means of the generalized Chow and Liu's algorithm (see Section 4). Finally, the partial solutions associated to each separator are joined together to form an M$(i+1)$DG. The procedure has a computational complexity of $\mathcal{O}(n^2 \log n)$, in the worst case. The divide-and-conquer strategy is repeated recursively $k-1$ times starting from the empty graph until an MkDG structure is reached.

3 A divide-and-conquer strategy

This section formally presents the intuitions behind the fractal tree algorithms. As we noted in the previous section, these algorithms are based on a two-fold divide-and-conquer strategy:

- To approach Problem 1 dividing it into $(k-1)$ Problems 2. This approach produces a sequence of MiDGs, for $i = 2, .., k$, each one coarser than the previous.
- To approach Problem 2 decomposing the input MkDG in terms of its separators and solving each separator (see Problem 3 and Algorithm 1).

The section starts by justifying that the strategy of constructing a sequence of coarser MiDGs, for $i = 2, ..., k$, can solve Problem 1. Then, we present the intuitions for obtaining an M$(k+1)$DG, given a thinner MkDG, based on the separators of the MkDG.

Proposition 1. *(Lemma 2.21, [10]) Let $\boldsymbol{G}$ be a decomposable graph and let $\boldsymbol{G}^-$ be a thinner decomposable graph with exactly m edges less. Then, there is a sequence of coarser structures $\boldsymbol{G}^- = \boldsymbol{G}^0 \prec \boldsymbol{G}^1 \prec ... \prec \boldsymbol{G}^m = \boldsymbol{G}$ of decomposable graphs that differs exactly in one edge.*

According to this result, any DG can be obtained starting from a thinner DG following a sequence of DGs which differ exactly in one (candidate) edge. Therefore, we can construct a sequence of coarser MiDGs, for $i = 2, ..., k$, to attain any target MkDG.

Corollary 3. *Let $\boldsymbol{G}_k$ be an MkDG. It is possible to construct a sequence of MiDG structures, $\boldsymbol{G}_i$, for $i = 2, ..., k$, where $\boldsymbol{G}_2 \prec \boldsymbol{G}_3 \prec ... \prec \boldsymbol{G}_{k-1} \prec \boldsymbol{G}_k$.*

Proof. It is a direct consequence of Definition 4 and Proposition 1. □

This sequence of coarser structures is the basis for the algorithms proposed in Section 5. Since all the intermediate structures are DGs, the maximum likelihood parameters can be obtained in a closed form and, as we noted in Section 2.2, we can focus on the structural learning only.

Next, we present the intuitions for obtaining an M$(k+1)$DG from a thinner MkDG following a divide-and-conquer strategy.

In order to obtain any M$(k+1)$DG, $\boldsymbol{G}_{k+1}$, from a thinner MkDG, $\boldsymbol{G}_k$, we are interested in creating cliques of size $k+1$ maintaining the decomposability of the graph. By Theorem 1, in order to create a clique of size $k+1$ maintaining the decomposability, we require to add a (candidate) edge $\{u, v\}$, where u and v

are minimally separated by a separator S of size $k-1$ and $\{u,v\} \subseteq \mathbb{N}(S)$ (see Observation 1). Thus, we have to identify all the separators of size $k-1$ for $\boldsymbol{G}_k$, and for any $(k+1)$DG coarser than $\boldsymbol{G}_k$. The following results show that the set of separators of all these structures is a subset of the set of separators of $\boldsymbol{G}_k$.

Proposition 2. *Let $\boldsymbol{G}_k$ be an MkDG and let $\boldsymbol{G}^+$ be a coarser $(k+1)$DG. Let $\boldsymbol{G}^-$ be a DG coarser than (or equal to) $\boldsymbol{G}_k$ and thinner than $\boldsymbol{G}^+$, where $\boldsymbol{G}^-$ has one edge less than $\boldsymbol{G}^+$. The set of the minimal separators of size $k-1$ of $\boldsymbol{G}^+$ is contained in the set of minimal separators of size $k-1$ of $\boldsymbol{G}^-$.*

Proof. By definition of MkDG, $\boldsymbol{G}^-$ has cliques of size k or $k+1$, only. Since $\boldsymbol{G}^+$ is a $(k+1)$DG coarser than an MkDG, it must be obtained from $\boldsymbol{G}^-$ by adding a candidate edge $\{u,v\} \in \mathbb{E}_{k+1}(\boldsymbol{G}^-)$. Thus, there exists a chain of cliques $C_1, ..., C_i, C_{i+1}, ..., C_m$ for $\boldsymbol{G}^-$ with the sequence of separators $S_2, ..., S_{i+1}, ..., S_m$, where $|S_{i+1}| = k-1$, $u \in C_i \setminus S_{i+1}$ and $v \in C_{i+1} \setminus S_{i+1}$.

There are three possible situations depending on the sizes of C_i and C_{i+1}. First, if $|C_i| = |C_{i+1}| = k$ then, the addition of $\{u,v\}$ replaces the cliques C_i and C_{i+1} by a new clique $S_{i+1} \cup \{u,v\}$ and removes S_{i+1} from the sequence of separators. Secondly, if $|C_i| = k$ and $|C_{i+1}| = k+1$ then, the clique C_i is replaced by $S_{i+1} \cup \{u,v\}$ and the separator S_i by $S_i \cup \{v\}$. And thirdly, if $|C_i| = |C_{i+1}| = k+1$ then, the addition of $\{u,v\}$ generates the new clique $S_{i+1} \cup \{u,v\}$ and replaces the separator S_{i+1} by the separators $S_{i+1} \cup \{u\}$ and $S_{i+1} \cup \{v\}$.

In the three cases the separator S_i is removed or replaced by separators of size greater than $k-1$. Therefore, the set $\{S \in \mathbb{S}(\boldsymbol{G}_k^+) : |S| = k-1\} \subseteq \{S \in \mathbb{S}(\boldsymbol{G}_k^-) : |S| = k-1\}$. □

Thus, the addition of a candidate edge to an MkDG or a $(k+1)$DG coarser than an MkDG reduces or at least maintains its set of minimal separators of size $k-1$. In other words, a new separator of size $k-1$ can not be created by the addition of a candidate edge. For example, the addition of $\{2,4\}$ to $\boldsymbol{G}_3$ to create $\boldsymbol{G}^+$, removes the separator $\{1,5\}$.

Corollary 4. *Let $\boldsymbol{G}_k$ be an MkDG and let $\boldsymbol{G}^+$ be a coarser DG. The subset of minimal separators of $\boldsymbol{G}^+$ of size $k-1$ is contained in the set of separators of $\boldsymbol{G}_k$.*

Proof. It is a direct consequence of Proposition 2. □

Consequently, given the MkDG, $\boldsymbol{G}_k$, the set of separators $\mathbb{S}(\boldsymbol{G}_k)$ contains all the relevant separators for identifying the edges required to learn a coarser M$(k+1)$DG.

The following results strengthen the idea of using a divide-and-conquer strategy, focused on the mantles of separators of size $k-1$, for learning an M$(k+1)$DG

from a thinner MkDG. We start by characterizing the mantles of the separators of size $k-1$.

Corollary 5. *Let $\boldsymbol{G}_k$ be an MkDG and let $\boldsymbol{G}^+$ be a $(k+1)$DG coarser than $\boldsymbol{G}_k$.*

1. *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.*
2. *The mantle of any $S \in \mathbb{S}(\boldsymbol{G}_k)$ in $\boldsymbol{G}^+$, $\boldsymbol{G}^+(\mathbb{N}(S))$, is a forest.*

Proof. All the cliques in $\boldsymbol{G}_k$ are of size k and the separators of size $k-1$. If the mantle of a separator $S \in \mathbb{S}(\boldsymbol{G}_k)$ has an edge $\{u, v\}$, then $\boldsymbol{G}_k$ has the clique $\{u, v\} \cup S$ of size $k+1$. Since $\boldsymbol{G}_k$ is an MkDG, the mantle is an empty graph.

If the mantle of any set $S \in \mathbb{S}(\boldsymbol{G}_k)$ in $\boldsymbol{G}^+$ has a clique C greater than 2, then $\boldsymbol{G}_k$ has the clique $C \cup S$ of size greater than $k+1$. Since $\boldsymbol{G}^+$ is an $(k+1)$DG, the mantle is a forest. □

For example, the mantles of $\{1, 5\}$ and $\{2, 5\}$ in $\boldsymbol{G}_3$ are the empty graphs $(\{2, 4\}, \emptyset)$ and $(\{1, 3, 6\}, \emptyset)$, while in $\boldsymbol{G}^+$ they are the forests $(\{2, 4\}, \{\{2, 4\}\})$ and $(\{1, 3, 4, 6\}, \{\{1, 4\}\})$, respectively (See Figure 2).

Next, we provide an appropriate characterization of the set of candidate edges associated to a set of vertices of size $k-1$. It should be noted that this set consists of the candidate edges that create cliques of size $k+1$, only.

Definition 5. *Let S be a subset of vertices that induces a complete graph in the DG $\boldsymbol{G}$. The set of candidate edges associated to S in $\boldsymbol{G}$ is defined as*

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

Thus, the set of candidate edges associated to a separator is defined as the set of candidate edges which creates cliques of size 2 in its mantle. In other words, the set of candidate edges associated to a separator are given by pair of vertices which belong to different connected components of the mantle, without considering the particular edges included in the mantle. For example, the set of candidate edges for $\{2, 5\}$ in $\boldsymbol{G}_3$ is $\mathbb{E}(\{2, 5\}|\boldsymbol{G}_3) = \{\{1, 3\}, \{1, 6\}, \{3, 6\}\}$, while in $\boldsymbol{G}^+$ it is $\mathbb{E}(\{2, 5\}|\boldsymbol{G}_3) = \{\{1, 3\}, \{1, 6\}, \{3, 6\}, \{3, 4\}, \{4, 6\}\}$.

Based on the previous definition, we characterize the set of vertices that create cliques of size $k+1$.

Proposition 3. *Let $\boldsymbol{G} = (V, E)$ be an MkDG or a $(k+1)$DG coarser than an MkDG. Then,*

1. $\mathbb{E}_{k+1}(\boldsymbol{G}) = \cup_{S \in \mathbb{S}(\boldsymbol{G}):|S|=k-1} \mathbb{E}(S|\boldsymbol{G})$, *and*

2. *for any S and S' in $\mathbb{S}(\boldsymbol{G})$, where $S \neq S'$, $\mathbb{E}(S|\boldsymbol{G}) \cap \mathbb{E}(S'|\boldsymbol{G}) = \emptyset$.*

Proof. The first point is a direct consequence of Theorem 1 and the fact that $\mathbb{E}_{k+1}(\boldsymbol{G})$ consists of the candidate edges that create cliques of size smaller or equal to $k+1$. The second point is proved by the contradiction method. Let us suppose a candidate edge $\{u, v\} \in \mathbb{E}(S|\boldsymbol{G}) \cap \mathbb{E}(S'|\boldsymbol{G})$. Then, u and v must be separated by two distinct minimal separators. Therefore, by Theorem 1, $\{u, v\}$ can not be a candidate edge, which contradicts the starting assumption. □

In summary, the set of candidate edges which create cliques of size $k+1$, can be obtained by a local inspection of the separators of size $k-1$. Additionally, the set of candidate edges associated to different separators are disjoint. For example the set of candidate edges for $\boldsymbol{G}_3$ which creates cliques of size 4 is $\mathbb{E}_4(\boldsymbol{G}_3) = \{\{2,4\},\{1,3\},\{1,6\},\{3,6\}\}$ where $\mathbb{E}(\{1,5\}|\boldsymbol{G}_3) = \{\{2,4\}\}$ and $\mathbb{E}(\{2,5\}|\boldsymbol{G}_3) = \{\{1,3\},\{1,6\},\{3,6\}\}$. And the set of candidate edges for $\boldsymbol{G}^+$ is $\mathbb{E}(\boldsymbol{G}^+) = \{\{1,3\},\{1,6\},\{3,6\},\{3,4\},\{4,6\}\}$ where $\mathbb{E}(\{1,5\}|\boldsymbol{G}^+) = \emptyset$ and $\mathbb{E}(\{2,5\}|\boldsymbol{G}^+) = \mathbb{E}(\boldsymbol{G}^+)$.

Proposition 4. *Let $\boldsymbol{G}_k = (V, E)$ be an MkDG and let $\boldsymbol{G}^+$ be a $(k+1)$DG coarser than $\boldsymbol{G}_k$. Then, $\boldsymbol{G}^+$ is an M$(k+1)$DG if and only if for any $S \in \mathbb{S}(\boldsymbol{G}_k)$ we have that $\boldsymbol{G}^+(\mathbb{N}(S))$ is a tree.*

Proof. By Corollary 1, $\mathbb{E}_{k+1}(\boldsymbol{G}^+)$ must be the empty set. By Corollary 4 and Proposition 3, $\mathbb{E}_{k+1}(\boldsymbol{G}^+)$ is empty if and only if for any $S \in \mathbb{S}(\boldsymbol{G}_k)$ we have that $\mathbb{E}(S|\boldsymbol{G}^+)$ is empty. Thus, for any $S \in \mathbb{S}(\boldsymbol{G})$, $\boldsymbol{G}^+(\mathbb{N}(S))$ must have a single connected component and by Corollary 5 is a tree. □

This proposition states that Problem 2, learning a maximum likelihood M$(k+1)$DG from a thinner MkDG, $\boldsymbol{G}_k$, seems to be solved by learning maximum likelihood trees for the mantles of each separator of $\boldsymbol{G}_k$ independently. We call the learning of a maximum likelihood tree for the mantle of a separator the separator problem (see Problem 3) and it is efficiently solved by the Generalized Chow and Liu's algorithm proposed in Section 4. Unfortunately, Problem 2 is an NP-hard problem. It should be noted that the addition of a candidate edge $\{u, v\}$ associated to a separator can increase the neighborhood of other separators and, at the same time, it can increase their set of candidate edges. Thus, the order in which the edges are added can considerably change the structures that can be constructed due to the interaction among the mantles of adjacent separators, i.e., two separators of size $k-1$ are adjacent when they share $k-1$ vertices. Next, we show how the addition of an edge to the mantle of a separator can affect the mantles of adjacent separators.

Proposition 5. *Let $\boldsymbol{G}_k$ be an MkDG and let $\boldsymbol{G} = (V, E)$ be a DG coarser than or equal to $\boldsymbol{G}_k$ and thinner than an M$(k+1)$DG. Let S and S' be two distinct separators in $\mathbb{S}(\boldsymbol{G})$ and let u, v be a candidate edge associated to S. Let us denote $(V, E \cup \{u, v\})$ by $\boldsymbol{G}^+$, and $\boldsymbol{G}(\mathbb{N}(S'))$ by $(V_{S'}, E_{S'})$.*

If $S' \setminus S = \{u\}$ and $S \setminus S' = \{w\}$ then

1. *$\boldsymbol{G}^+(\mathbb{N}(S')) = (V_{S'} \cup \{v\}, E_{S'} \cup \{v, w\})$, and*
2. *$\mathbb{E}(S'|\boldsymbol{G}^+) = \mathbb{E}(S'|\boldsymbol{G}) \cup \{\{v, y\} : y$ and w are at distinct components in $(V_{S'}, E_{S'})\}$.*

Proof. $\boldsymbol{G}^+$ is obtained by adding $\{u, v\}$ to $\boldsymbol{G}$, which creates the clique $C = S \cup \{u, v\} = S' \cup \{v, w\}$. This clique belongs to the neighborhood of S' since $S' \subset C$. Thus, v belongs to the vertices of $\boldsymbol{G}^+(\mathbb{N}(S'))$ and $\{v, w\}$ to its set of edges.

Besides, v is in the same connected component of w in $\boldsymbol{G}^+(\mathbb{N}(S'))$ and, by Proposition 3 the set of edges $\{\{v, y\} : y$ and w are at distinct components in $(V_{S'}, E_{S'})\}$ is added to the set of candidate edges of S'. □

For example, the addition of the candidate edge $\{2, 4\}$ to $\boldsymbol{G}_3$ to create $\boldsymbol{G}^+$ increases the set of candidate edges, $\mathbb{E}_4(\boldsymbol{G}^+) \setminus \mathbb{E}_4(\boldsymbol{G}_3) = \{\{3, 4\}, \{4, 6\}\}$. Therefore, the addition of a candidate edge can cause the set of candidate edges associated to an adjacent separator to increase. This fact clearly indicates the importance of the order in which the edges are added and it states the difficulty of solving Problem 2. For example, the addition of $\{2, 4\}$ to $\boldsymbol{G}_3$ creates the candidate edges $\{3, 4\}$ and $\{4, 6\}$ which can be added to the mantle of $\{2, 5\}$ to create the M4DG $\boldsymbol{G}_4$, with the cliques $\{\{1, 2, 4, 5\}, \{1, 2, 3, 4\}, \{2, 4, 5, 6\}\}$. Instead of adding $\{2, 4\}$, if we first consider the addition of the candidate edge $\{3, 6\}$ then, $\boldsymbol{G}_4$ can not be attained.

At this point, we want to present a result which can be used to efficiently identify the separators of $\boldsymbol{G}_{k+1}$ given a thinner $\boldsymbol{G}_k$, by a local inspection of the mantles of separators $\mathbb{S}(\boldsymbol{G}_k)$. It is based on the degree of the nodes in the mantle of a separator.

Proposition 6. *Let $\boldsymbol{G}_k$ be an MkDG and let $\boldsymbol{G}_{k+1}$ an M$(k+1)$DG coarser than $\boldsymbol{G}_k$. The separators of $\boldsymbol{G}_{k+1}$ can be obtained from a local inspection of the mantles of $\mathbb{S}(\boldsymbol{G}_k)$ in $\boldsymbol{G}_{k+1}$ as follows:*

$$\mathbb{S}(\boldsymbol{G}_{k+1}) = \{S \cup \{u\} : S \in \mathbb{S}(\boldsymbol{G}_k), \text{ where } |\mathbb{N}(u|\boldsymbol{G}_{k+1}(\mathbb{N}(S|\boldsymbol{G}_{k+1})))| > 1\}$$

Proof. Let v and w be separated by S in $\boldsymbol{G}_k$ and let $\{u, v\}$ and $\{u, w\}$ be two edges of the mantle of S in $\boldsymbol{G}_{k+1}$. By Proposition 4, we know that the mantle $\boldsymbol{G}_{k+1}(\mathbb{N}(S))$ is a tree and thus, v and w are separated by u in the mantle. Therefore, $S \cup \{u\}$ separates v and w in $\boldsymbol{G}_{k+1}$. □

This result can be used by the fractal tree algorithms in order to efficiently obtain the set of separators required at each growing step. The same result can be used in order to provide an efficient implementation of the greedy procedures proposed by [13] for approaching Problem 1.

4 The separator problem and Generalized Chow and Liu's algorithm

In this section we formally define the separator problem. Next, we provide a natural extension of the Chow and Liu's algorithm (CL) called **generalized Chow and Liu** (**GCL**), which solves the separator problem with a computational complexity of $\mathcal{O}(n^2 \log n)$. The GCL algorithm is the building block of the fractal tree algorithms presented in Section 5. For a review of other extensions of the CL algorithm we refer the reader to ([7]).

Problem 3. ***[The separator problem]*** *Let $\boldsymbol{G}$ be an MkDG or a $(k+1)$DG coarser than an MkDG, where $\boldsymbol{G}$ has a single separator, S, of size $k-1$. Let $\mathcal{D}$ be a data set. Find an MkDG coarser than $\boldsymbol{G}$ which maximizes the likelihood.*

Note that for the particular case of an MkDG with a single separator, Problem 3 and Problem 2 become equivalent.

As we noted before, the maximization of the likelihood function is equivalent to the minimization of the entropy. The next proposition establishes the decrease in the entropy when a candidate edge is added to a DG:

Proposition 7. *Let $\boldsymbol{G} = (V, E)$ be a decomposable graph, let $e = \{u, v\}$ be a candidate edge associated to a separator S and let $\mathcal{D}$ be a data set. The entropy of $(V, E \cup \{e\})$ given $\mathcal{D}$ decreases in $\hat{I}(X_u, X_v|\boldsymbol{X}_S)$ with respect to $\boldsymbol{G}$.*

The GCL algorithm creates a maximum weighted tree in the neighborhood of S, where the weight of each candidate edge $\{u, v\} \in \mathbb{E}(S|\boldsymbol{G})$ is given by $\hat{I}(X_u, X_v|\boldsymbol{X}_S)$. This procedure is equivalent to the CL algorithm with the appropriate weights for dealing with Problem 3. The pseudo-code of the GCL procedure is given in Algorithm 1. The computational complexity of CGL is $\mathcal{O}((n-k)^2 \log(n-k))$, in the worst case due to the sorting of the $\binom{n-k}{2}$ candidate edges.

Corollary 6. *GCL solves Problem 3.*

Proof. By Corollary 4 and Proposition 4, the solution is the structure obtained by constructing a minimum entropy tree in the mantle of the unique separator of $\boldsymbol{G}$, S. Algorithm 1 learns a maximum likelihood tree for the mantle of S using the Kruskal algorithm for maximization, where, by Proposition 7, $\hat{I}(X_u, X_v|\boldsymbol{X}_S)$ is the weight for the edge $\{u, v\} \in \mathbb{E}(S|\boldsymbol{G})$. □

Algorithm 1. *(Generalized Chow and Liu's algorithm)*
Input: *An MkDG or a $(k+1)$DG coarser than an MkDG, $\boldsymbol{G}$, with a single separator, S, of size $k-1$, where $\boldsymbol{G}(\mathbb{N}(S)) = (V_S, E_S)$. A data set $\mathcal{D}$.*
Output: *A set of edges when added to $\boldsymbol{G}$ solves Problem 3.*
Pseudocode:

1. $E := \emptyset$
2. `For any` $\{u,v\} \in \mathbb{E}(S|\boldsymbol{G})$ `compute` $\hat{I}(X_u; X_v|\boldsymbol{X}_S)$ `(see Definition 5)`
3. `Sort` $\mathbb{E}(S|\boldsymbol{G})$ `in descending order of` $\hat{I}(X_u; X_v|\boldsymbol{X}_S)$
4. `While` $(V_S, E_S \cup E)$ `is not a tree:`
5. `Take the next edge` $\{u,v\} \in \mathbb{E}(S|\boldsymbol{G})$, `where` u `and` v `are at different connected components of` $(V_S, E_S \cup E)$
6. $E := E \cup \{u,v\}$
7. `return` E

The next result quantifies the (possibly huge) number of M$(k+1)$DGs that can be attained by the GCL algorithm.

Corollary 7. *Let $\boldsymbol{G}$ be an MkDG or a $(k+1)$DG coarser than an MkDG, which has a single minimal separator S of size $k-1$. Let the mantle of S be a forest with t connected components, each of them with n_i vertices for $i = 1, ..., t$. The number of M$(k+1)$DGs coarser than $\boldsymbol{G}$ is given by:*

$$\prod_{i=1}^{t} n_i \;\cdot\; t^{t-2-\sum_{i=1}^{t}(n_i-1)} \tag{4}$$

Proof. By Proposition 4, the number of M$(k+1)$DGs coarser than $\boldsymbol{G}$ is the number of trees coarser than $\boldsymbol{G}(\mathbb{N}(S))$. And by Lemma 6 in [12] this corresponds to Equation 4. □

The number of structures that can be obtained for the mantles shown in Figures 2b, 2d, 4b and 7a are $3^{3-2} = 3$, $2 \cdot 4^{4-2-1} = 8$, $4^{4-2} = 16$ and $2 \cdot 2 \cdot 6^{6-2-2} = 144$, respectively.

Finally, we show that the GCL algorithm finds one of the closest trees to a given structure when a sufficiently large data set is available.

Corollary 8. *Generalized Chow and Liu's algorithm is asymptotically consistent [4], i.e. $\lim_{N\to\infty} P(D_{KL}(p; p_{\boldsymbol{G}^+}) = D_{KL}(p; p_{\boldsymbol{G}^*})) = 1$, where $p_{\boldsymbol{G}^+}$ represents the model obtained by the GCL procedure, $p_{\boldsymbol{G}^*}$ is a maximum likelihood kDG coarser than $\boldsymbol{G}$, and D_{KL} represents the Kullback-Leibler divergence between two distributions.*

Proof. The proof is analogous to the proof of asymptotic consistency of the CL algorithm [4]. □

In summary, GCL inherits the theoretical properties of the CL algorithm:

- By Proposition 6, it solves Problem 3.
- GCL has a computational complexity in the worst case of $\mathcal{O}(n^2 \log n)$.
- By Corollary 8, GCL is asymptotically consistent.

5 Fractal tree algorithms

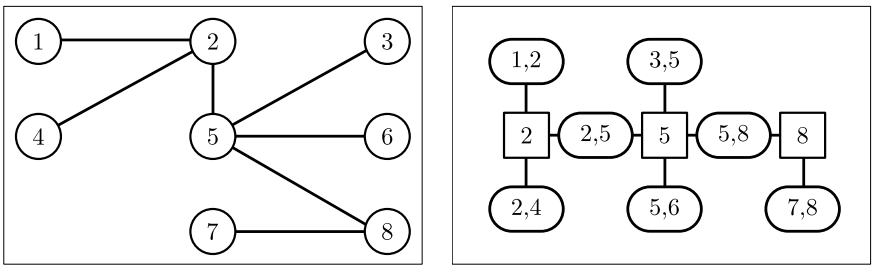
In essence, the proposed fractal tree algorithms approach Problem 1 in $k-1$ iterative growing steps. These steps create a sequence of MiDGs for $i = 2, ..., k$ each coarser than the previous one. Step i starts with an MiDG, $\boldsymbol{G}_i$, and learns an M$(i+1)$DG, $\boldsymbol{G}_{i+1}$ using a divide-and-conquer strategy based on the set of separators of $\boldsymbol{G}_i$.

The fractal tree algorithms make use of the structural features of MkDG structures which allow to approach Problem 2 efficiently:

- A maximum of $n - i$ separators have to be considered. In unconstrained kDGs the number of separators can be $\mathcal{O}(k \cdot n)$.
- The mantle associated to each separator, at the start of each growing step, is an empty subgraph. At the end of the growing step the mantles become trees. The problem of learning this tree (see Problem 3) is efficiently solved using the GCL algorithm (see Algorithm 1). The application of the GCL algorithm to each separator can be understood as a global operator in the sense of [6].
- All the separators are of size $k-1$ and, thus, the addition of candidate edges from $\mathbb{E}(S|\boldsymbol{G}_k)$ only generates cliques of size $k+1$. In consequence, every growing step effectively controls the size of all the created cliques.
- Efficient prune-and-graft procedures can be used (see Section 6). These procedures tend to improve the likelihood of the model by favoring the mobility of the (leaf) vertices.

In summary, due to the structural features of the MkDGs, the fractal tree algorithms have a computational complexity of $\mathcal{O}(k \cdot n^2 \log n)$ and can approach Problem 1 with efficacy. It should be highlighted that fractal tree algorithms learn a sequence of MkDGs and the user can choose the most appropriate value of k after the entire sequence is constructed, e.g. by means of the likelihood ratio statistical test.

Next, we propose two particular fractal tree algorithms: parallel fractal tree and sequential fractal tree. 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.



(a) Undirected graph representation of $\boldsymbol{G}_2$. (b) Juntion tree representation of $\boldsymbol{G}_2$.

Figure 3: 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 junction tree representation. In the junction tree representation, the cliques are represented by circles and the separators by squares. The lines connecting the cliques and separators represent the different separator problems. The junction tree representation highlights the relations among cliques and separators, removing the unnecessarily fine grain detail given by the undirected graph representation. The separators $\{2\}, \{5\}$ and $\{8\}$ shown in Figure 3b correspond to the vertices $2, 5$ and 8 of $\boldsymbol{G}_2$ with a degree higher than 1.

5.1 Parallel fractal tree

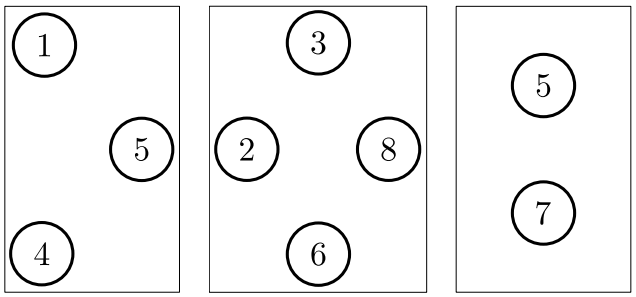
In this section we propose a tractable approximation to Problem 1 called **parallel fractal tree** (**PFT**). The pseudo-code of PFT is given in Algorithm 2. At each parallel growing step (lines 3-5), this algorithm solves the separator problems ignoring the interactions among the mantles of different separators (see Proposition 5). The interactions are taken into account once all the separator problems are solved (see line 5). 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 $\boldsymbol{G}_i$.

Algorithm 2. *(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 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 the GCL algorithm to the mantle of* S *for obtaining* E^S *(Algorithm 1).*
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* $\boldsymbol{G}_{i+1}$ *(Proposition 6).*
8. *Return* $\boldsymbol{G}_k$*.*

In PFT, the growing step (Algorithm 2, 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. The GCL algorithm can be also (partially) parallelized: the computation of the empirical conditional mutual information can be performed in parallel and the sorting of the edges can be partially parallelized. 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.

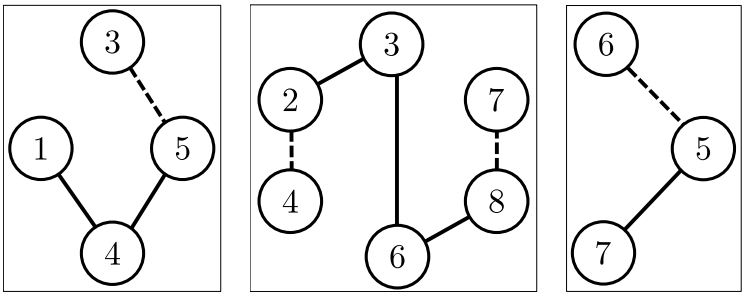


(a) $\boldsymbol{G}_2(\mathbb{N}(\{2\})$. (b) $\boldsymbol{G}_2(\mathbb{N}(\{5\})$. (c) $\boldsymbol{G}_2(\mathbb{N}(\{8\})$.

Figure 4: This figure shows the mantles of the separators of the M2DG, $\boldsymbol{G}_2$, shown in Figure 3. The mantles shown correspond to the separators $\mathbb{S}(\boldsymbol{G}_2) = \{\{2\}, \{5\}, \{8\}\}$. Note that all the mantles are empty subgraphs.

Figures 3, 4, 5 and 6 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 algorithm 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 3, into the mantles associated to the separators, $\{2\}$, $\{5\}$ and $\{8\}$. The corresponding mantles are shown in Figures 4a, 4b and 4c, respectively. Once the mantles are determined, the GCL algorithm is applied to

solve each separator problem without taking into account the interactions among them. The obtained results are illustrated in Figures 5a, 5b and 5c, respectively. Finally, the solutions to each separator problem are gathered together to form the attained M3DG, shown in Figure 6.



(a) The solution to $\boldsymbol{G}_2(\mathbb{N}(\{2\}), \boldsymbol{G}_3^{PFT}(\mathbb{N}(\{2\}))$.

(b) The solution to $\boldsymbol{G}_2(\mathbb{N}(\{5\}), \boldsymbol{G}_3^{PFT}(\mathbb{N}(\{5\}))$.

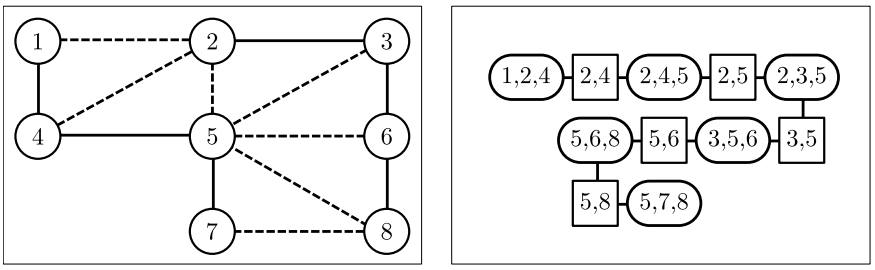
(c) The solution to $\boldsymbol{G}_2(\mathbb{N}(\{8\}), \boldsymbol{G}_3^{PFT}(\mathbb{N}(\{8\}))$.

Figure 5: This figure shows the solutions, given by the PFT algorithm, to the separator problems represented in Figure 4. Note that all the solutions correspond to trees. The solid lines represent the edges added during the parallel growing step to each mantle by the GCL algorithm while the dashed lines represents the edges that appear in the mantles due to their interactions.

At step i, the PFT algorithm can explore a (possibly) huge space of candidate M$(i+1)$DGs. The search space of PFT at step i is given by the number of vertices in the mantle of each separator. Note that, by Corollary 5, all the mantles of an MiDG are empty subgraphs. As a consequence of Corollary 7, the number of candidate M$(i+1)$DGs attainable from a given MiDG is $\sum_{S \in \mathbb{S}(\boldsymbol{G}_i)} |\mathbb{N}(S)|^{|\mathbb{N}(S)|-2}$. For example, the number of candidates for the graph represented in Figure 3 is $3^1 \cdot 4^2 \cdot 2^0 = 144$.

The computational complexity of PFT in the worst case is $\mathcal{O}(\prod_{i=1}^{k-1}(n-i+1)^2 \log(n-i+1))$. Roughly speaking, when $n >> k$, the computational complexity is $\mathcal{O}(k \cdot n^2 \log n)$, that is, linear with respect to the order of the obtained decomposable model and quadratic with respect to the number of variables. 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 the performed experimentation we have observed a tendency to achieve chain structures (see Figure 6). A chain structure limits the set of attainable structures by a parallel growing step (lines 3-4, Algorithm 2) to one, because all the mantles have only two vertices. Besides, once a chain structure is reached, the



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

Figure 6: 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 5). The solid lines in Figure 6a represent the edges added in the last growing step while the dashed lines represent the edges added in the previous step.

next parallel growing steps will produce chain structures. We call this phenomena the **chain caveat**. The convergence to chain structures is a direct consequence of the expected degree of the vertex of a tree, which is 2. Fortunately, the expected number of vertices with degree d in a tree seems to be proportional to 2^{-d} [19].

PFT can not attain decomposable graphs with cliques containing more than two separators. This is due to the fact that each parallel growing step adds a forest (see Figure 6a). 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 8, where the clique $\{2, 4, 5\}$ contains the separators $\{2, 4\}$, $\{2, 5\}$, and $\{4, 5\}$. We call this phenomena 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.

The next section introduces the sequential fractal tree algorithm which avoids the chain and the unattainable structure caveats.

5.2 Sequential fractal tree algorithm

This section proposes the **sequential fractal tree** (**SFT**) algorithm. SFT increases the number of candidate edges considered at each growing step with respect to PFT and performs a prune-and-graft procedure to the leaf vertices, i.e. vertices that belong to a single separator. 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 or higher to that obtained by PFT. The pseudocode of SFT is shown in Algorithm 3.

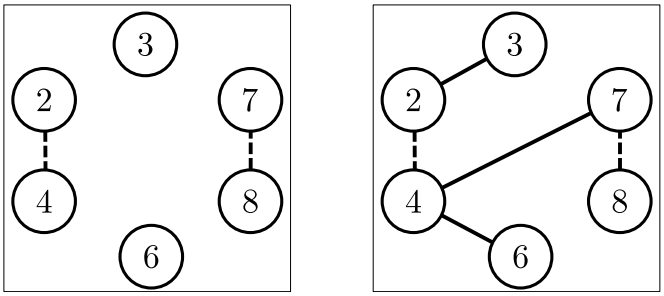
Algorithm 3. *(Sequential fractal tree)*
Input: *A data set $\mathcal{D} = \{\boldsymbol{x}^1, ..., \boldsymbol{x}^N\}$, a positive integer k.*
Output: *An MkDG.*
Pseudocode:

```
1. Construct 𝕊(G_1) for the empty graph G_1.
2. For i = 1, ..., k − 1:
3.    Sort the separators in 𝕊(G_i) for G_i, e.g. by the number of connected
   components of their associated subgraph.
4.    G_{i+1} := G_i.
5.    For S ∈ 𝕊(G_i) in order do (sequential growing step):
6.      Apply the GCL algorithm to the mantle of S in G_{i+1} for obtaining
   E (Algorithm 1).
7.      Add E to G_{i+1} (Proposition 5).
8.    Obtain the separators of G_{i+1} (Proposition 6).
9. Return 𝕊(G_k) for G_k.
```

In SFT, the separators are solved sequentially using the GCL algorithm (Algorithm 3, lines 4-6). After the application of GCL to a separator, the mantles of the adjacent separators are modified (Proposition 5) 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 obtained (see Corollary 7). For example, the mantle of separator $\{5\}$ considered by SFT in its 2nd growing step, compared to the mantle considered by PFT, takes into account the set of additional candidate edges $\{\{3,4\},\{3,7\},\{4,6\},\{4,7\},\{4,8\},\{6,7\}\}$ (see Figures 7a and 4b). The number of possible solutions for the mantle of $\{5\}$ in SFT (Figure 7a) is $2 \cdot 26^{6-2-2} = 144$; in PFT (Figure 4b) the number is $4^2 = 16$. Moreover, all the structures that can be attained by the parallel growing procedure of PFT can be attained by the sequential growing procedure of SFT. It should be noted that, while the parallel growing step adds forests to the input MkDG, the sequential growing can add more general structures, which avoids the unattainable structure caveat. For example, Figure 8 shows a structure where the clique $\{2,4,5\}$ contains three separators, $\{2,4\}$, $\{2,5\}$ and $\{4,5\}$.

The sequential growing step requires to define 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 components of their mantles in ascending order. This

heuristic tends to maximize the number of MkDG structures that can be attained by the growing step. Moreover, in the performed experimentation we have observed that this criteria tends to obtain the best results for approaching Problem 1.

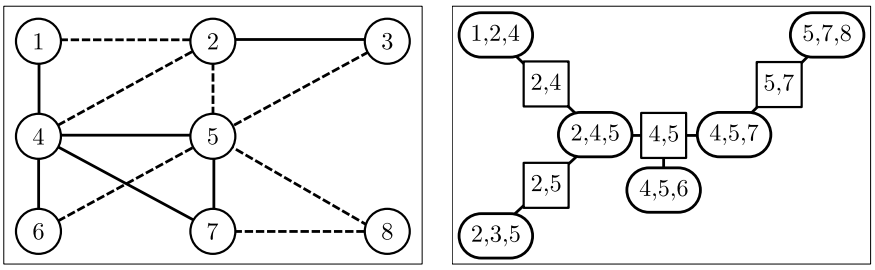


(a) $\boldsymbol{G}^{+}(\mathbb{N}(\{5\})$. (b) The solution to $\boldsymbol{G}^{+}(\mathbb{N}(\{5\})$.

Figure 7: This figure shows the mantle of 5 in the graph $\boldsymbol{G}^{+}$ obtained from $\boldsymbol{G}_2$ after the addition of the edges $\{2, 4\}$ and $\{7, 8\}$.

Figures 3, 4, 5, 7 and 8 illustrate an execution of SFT in a domain with $n = 8$ random variables for a maximum clique size of $k = 3$. The first ($i = 1$) sequential growing step (lines 4-6, Algorithm 3) is equivalent to PFT and the obtained structure is shown in Figure 3. 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 4a, 4b and 4c. 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 5c). However, the addition of the edge $\{5, 7\}$ interacts with the mantle of separator $\{5\}$, as $\{5\} \subset \{8\} \cup \{5, 7\}$ (see Proposition 5). 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 5a). In this case, the addition of the edge $\{4, 5\}$ modifies the mantle of separator $\{5\}$, as $\{5\} \subset \{8\} \cup \{4, 5\}$. Finally, the mantle of the separator $\{5\}$ is solved. Its mantle has changed due to the addition of edges in previous mantles (see Figure 7a). The solution is shown in Figure 7b. 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 8.

SFT has the same computational complexity as PFT. The growing step i is



(a) Undirected representation of $\boldsymbol{G}_3^{SFT}$. (b) Junction tree representation of $\boldsymbol{G}_3^{SFT}$.

Figure 8: This figure shows the solution obtained by PFT, $\boldsymbol{G}_3^{SFT}$, which has been created by adding the partial solutions to the mantles of each separator of $\boldsymbol{G}_2$ (see Figures 4a, 7a and 4c). The solid lines represent the edges added in the last growing step while the dashed lines represent the edges added in the previous step.

$\mathcal{O}((n-i)^2 \log(n-i))$, in its worst case. In summary, SFT has a computational complexity of $\mathcal{O}(\sum_{i=1}^{k-1}(n-i)^2 \log(n-i))$. When n and k are not related being $n >> k$, the computational complexity is $\mathcal{O}(k \cdot n^2 \log n)$. At this point we want to indicate that the sequential growing step can be partially parallelized taking into account the order of the separators.

6 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 pruning each leaf vertex from its separator for grafting in the separator which 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. The computational complexity of this procedure is $\mathcal{O}(n^2)$, in the worst case.

Next, we define the terms leaf (vertex) and stem (separator) in order to simplify the description of the P&G procedure.

Definition 6. *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 the number of leaf vertices in its separator is the size of its mantle minus one, $|\mathbb{N}(S) \setminus \bigcup \mathbb{S}(\boldsymbol{G}_k)| = |\mathbb{N}(S)| - 1$.*

In other words, a leaf of an MkDG belongs to a single clique C, which contains

a single separator $S \in \mathbb{S}(\boldsymbol{G}_k)$. A leaf can be removed from a MkDG maintaining the decomposability. 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 by effect of the P&G procedure and, thus, it can be removed from $\mathbb{S}(\boldsymbol{G}_k)$.

Algorithm 4. *(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. $S^* := \arg_{S' \in \mathbb{S}(\boldsymbol{G}_k)} \max I(X_u; \boldsymbol{X}_{S'})$
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. $S^* := \arg_{S' \in \mathbb{S}(\boldsymbol{G}_k)} \max I(X_u; \boldsymbol{X}_{S'})$
21. $E := (E \setminus \{\{u, v\} : v \in S\}) \cup \{\{u, w\} : w \in S^*\}$
22. return (V, E)

The pseudo-code of P&G is given in Algorithm 4. The P&G procedure is divided in two parts. In the first part (lines 1-17), 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 S consists of removing the $k-1$ edges $\{\{u, v\} : v \in S\}$. The pruned leaf can be grafted in any separator S'. The grafting of u in a separator S' is the addition of the $k-1$ edges $\{\{u, w\} : w \in S'\}$. In other words, the whole process replaces the clique $\{v\} \cup S$ by the clique $\{v\} \cup S'$ and produces an MkDG (lines 10,21).

The pruning of u from the separator S, increases the entropy of the structure in $\hat{I}(X_u; \boldsymbol{X}_S)$. To graft u in S' reduces the entropy of the graph in $\hat{I}(X_u; \boldsymbol{X}_{S'})$. Thus, the optimal pruning and grafting of leaf vertex u consists of pruning from its separator S and grafting into the separator $S^* = \arg_{S' \in \mathbb{S}(\boldsymbol{G}_k)} \max \hat{I}(X_u; \boldsymbol{X}_{S'})$ (line 9,20). Note that, when $S = S^*$ the prune-and-graft process has no effect over the MkDG.

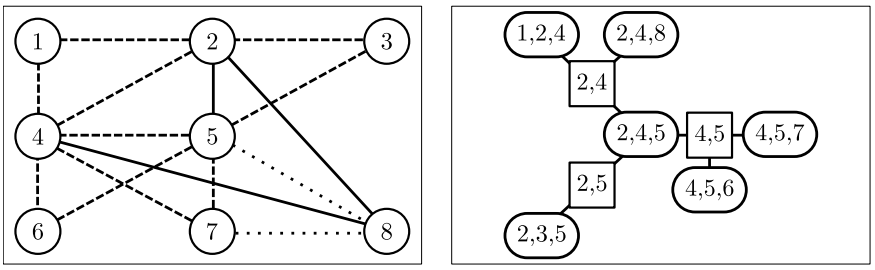
P&G has the following interesting properties:

- It transforms an MkDG into another MkDG with an equal or higher likelihood.
- The prune-and-graft favors the mobility of leaves across the entire MkDG structure.

Figures 8 and 9 illustrate the effect of the P&G procedure (Algorithm 4). P&G takes the M3DG $\boldsymbol{G}_3^{SFT}$ shown in Figure 8 as input and identifies the stems $\{2, 4\}$, $\{2, 5\}$ and $\{4, 5\}$. Then, considers the pruning and grafting of their leaves $\{1, 3, 8\}$. In this case, P&G only prunes the leaf vertex 8 from the neighborhood of $\{5, 7\}$ and grafts it in the separator $\{2, 4\}$. Since the neighborhood of $\{5, 7\}$ has 4 as its only vertex, the separator is removed and the separator $\{4, 5\}$ becomes a new stem with the leaves $\{6, 7\}$ (see Figure 9).

At the end of the first step the prune-and-graft operator is not applied because the obtained M2DG is optimal from the likelihood point of view (line 8, Algorithm 3).

In the worst case, there are $\mathcal{O}(n)$ leaf vertices and $\mathcal{O}(n)$ separators. Each leaf vertex is considered once. 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 $\mathcal{O}(n^2)$. The P&G procedure is applied $k-2$ times in fractal tree algorithms. It should be noted that the M2DG obtained at the end of the first growing step of the fractal tree algorithms is optimal



(a) Undirected graph representation (b) Junction tree representation

Figure 9: This figure shows the structure obtained after the prune-and-graft operator is applied to $\boldsymbol{G}_3^{SFT}$ (see Figure 8). The stems of $\boldsymbol{G}_3^{SFT}$ are the separators $\{2,4\},\{2,5\}$ and $\{5,7\}$. The leaves corresponding to the separators $\{2,4\},\{2,5\},\{4,5\}$ and $\{5,7\}$ are $1,3,6$ and 8, respectively. 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. Note that, at the end of the procedure the stem $\{5,7\}$ has been removed from the structure and the separator $\{4,5\}$ is a new stem.

(CL algorithm) and the P&G procedure has no effect in that structure. Thus, the computational complexity of the fractal tree algorithms using the P&G procedure remains $\mathcal{O}(k \cdot n^2 \log n)$.

7 Experimentation

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

- **Power of fitness**: 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 to compare results obtained for a different number of random variables and number of samples. The score is negative and a higher value 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 fitness. 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 fitness tend to the same value: This property inspires the next score.

- **Degree of overfitting**: Quantifies the overfitting phenomena of the model. It is calculated as the difference of the power of fitness minus the power of generalization, divided by the power of fitness. It can be interpreted as the percentage in which the power of fitness overestimates the power of generalization. A lower value of the score represents a better result. A negative or zero degree of overfitting indicates absence of overfitting.

- **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. The experiments have been carried out in a cluster composed of Intel-Xeon X5650 at 2.67GHz.

It should be noted that these scores allow to perform an intuitive scalability study with respect to the number of random variables and the number of training samples.

PFT and SFT are compared against the **forward greedy** algorithm (**FG**) proposed by [13]. In order to deal with domains of high dimensionality, we have implemented an efficient procedure based on a list of mantles of the separators of size lower than k, following the intuitions provided in Section 3. The efficient implementation of FG shares many similarities with the fast implementation proposed in [5].

Additional experiments were carried out with the bounded treewidth Bayesian network algorithm (BTBN) proposed in [6]. The treewidth is closely related to the maximum clique size of decomposable models and by limiting its value we can effectively control the number of computations required to perform probabilistic inference tasks. As far as we know, BTBN is one of the most efficient algorithms proposed in the literature capable of dealing with Problem 1. Even if BTBN has a computational complexity polynomial in the number of random variables, the computational complexity is still too high compared to the fractal tree algorithms. As a consequence, the time required to obtain results in most of the domains of the experimentation is prohibitive due to their high dimensionality. In practice, the high computational requirements of BTBN is mainly due to the InducedNodes procedure (Algorithm 4, [6]). This procedure is called many times by BTBN in order to

find an optimal chain of edges. Besides, InducedTime performs a (possibly) high number of maximum cardinality searches each time it is called. In the performed additional experiments BTBN with $n = 20, 25, 30, 25, 40$ and $k = 3, 4, 5$ has obtained worse results than SFT and similar results to PFT. As k increases, BTBN obtained worse results than SFT. The execution time of BTBN grows very fast as n increases. Figure 11 in Section 7.2 includes the results of BTBN for artificial domains with $n = 40$.

We have included the CL algorithm in the comparison as a reference for the following reasons:

- 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.

- We can compare the execution time of the proposed methods with respect to CL, which is the lower bound for the three algorithms.

Next, we present the obtained results grouped by the type domains used: real-world domains and artificial domains.

7.1 Real domains

This section presents a set of results obtained in 33 real domains from the UCI repository. The main features of these domains (n and N) are summarized in Table 1. The continuous random variables have been discretized in two intervals by means of the equal frequency strategy. Next, the missing values have been inputted using 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. In the experimentation we have only included data sets with a minimum of 20 random variables. The experiments have been performed for $k \in \{3, 4, 5\}$.

Tables 2 and 3 show the obtained results. For each data set, the table includes the power of fitness and the overfitting degree of CL, FG, PFT and SFT. The power of fitness and the overfitting degree are represented by bigger and smaller fonts, respectively. The results have been obtained using a 10-fold cross-validation estimator. Table 3 includes the average power of fitness (line Avg.) and the average rank (line Rank) across all the domains. The average ranks have been calculated independently for each k value. The best average powers of fitting and average ranks for each k value are shown in bold.

Next, we present the main conclusions that can be obtained from the experimentation in the UCI data sets:

Id	Name	N	n	ID.	Name	N	n
1	arrhythmia	452	278	18	KDD: IPUMS la 97	7019	37
2	audiology	226	68	19	KDD: IPUMS la 98	7485	37
3	autos	205	25	20	KDD: IPUMS la 99	8844	36
4	Cylinder bands	540	36	21	KDD: control chart time series	600	61
5	dermatology	366	35	22	kr vs kp	3196	37
6	Digit: Fourier coefficients	2000	77	23	Large soybean	683	35
7	Digit: Karhunen-Love coefficients	2000	65	24	lung cancer	32	57
8	Digit: pixel averages	2000	241	25	Marine Sponges	76	45
9	Digit: profile correlations	2000	217	26	mushroom	8124	22
10	Digit: Zernike moments	2000	48	27	Optical recognition of handwritten digits	5620	65
11	E. coli: promoter gene sequences	106	58	28	Primate splice junction gene sequences	3190	61
12	Flags	194	27	29	Sonar: mines VS rocks	208	61
13	Hepatitis	155	20	30	Soybean	683	35
14	Image segmentation	2310	20	31	SPAM e-mail	4601	58
15	Ionosphere	351	35	32	SPECT heart	267	23
16	IRAS Low resolution spectrometer	531	102	33	Waveform	5000	41
17	KDD: internet usage	10108	66				

Table 1: 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).

- SFT has obtained the best average power of fitness values and PFT the worst, for the different values of k (see the average ranking values in Table 3). On average, SFT and FG obtain similar power of fitting values (see average power of fitting in Table 3).
- PFT shows the lowest overfitting values (see the average ranking values 3). This result suggests that PFT learns models with lower variance because its search is more constrained than FG and SFT. PFT could be the most appropriate approach when the size of the available samples is small.
- **CL has obtained competitive results in many domains and in some of these domains FG, PFT and SFT obtained high overfitting degrees (**$Id = 2, 3, 4, 5, 8, 11, 12, 13, 24$**), especially with** $k = 5$**. This explains why FG, PFT and SPT obtained similar results in most of the domains.**

7.2 Artificial domains with MkDG structure

This section presents a set of results obtained in domains with MkDG structure. These structures have been randomly obtained using the procedure described in Algorithm 5. We have generated structures for $k \in \{3, 4, 5\}$ and $n \in \{40, 80, 160, 320, 640, 1280\}$. Note that for these parameters, the probability of obtaining an unattainable struc-

Id		$k=3$			$k=4$			$k=5$		
	CL	FG	PFT	SFT	FG	PFT	SFT	FG	PFT	SFT
1	-0,368	-0,349	-0,358	-0,349	-0,335	-0,346	-0,336	-0,32	-0,333	-0,321
	0,005	0,063	0,059	0,066	0,101	0,087	0,104	0,169	0,138	0,174
2	-0,248	-0,218	-0,226	-0,218	-0,186	-0,202	-0,191	-0,159	-0,179	-0,165
	0,056	0,495	0,385	0,454	1,038	0,777	0,995	1,761	1,369	1,818
3	-0,706	-0,582	-0,614	-0,584	-0,484	-0,527	-0,478	-0,409	-0,459	-0,4
	0,040	0,337	0,261	0,312	0,667	0,499	0,674	1,071	0,771	1,088
4	-0,692	-0,632	-0,64	-0,63	-0,58	-0,594	-0,579	-0,531	-0,548	-0,53
	0,017	0,141	0,133	0,149	0,279	0,258	0,292	0,458	0,434	0,472
5	-0,797	-0,712	-0,716	-0,711	-0,613	-0,63	-0,611	-0,49	-0,521	-0,481
	0,038	0,284	0,275	0,287	0,728	0,652	0,738	1,553	1,338	1,630
6	-0,888	-0,834	-0,837	-0,832	-0,799	-0,809	-0,798	-0,763	-0,777	-0,762
	0,000	0,024	0,024	0,024	0,045	0,044	0,045	0,084	0,077	0,083
7	-0,9	-0,864	-0,864	-0,864	-0,838	-0,842	-0,839	-0,809	-0,815	-0,808
	0,000	0,027	0,028	0,028	0,050	0,046	0,049	0,084	0,080	0,087
8	-1,078	-0,893	-0,938	-0,912	-0,718	-0,841	-0,786	-0,535	-0,744	-0,654
	0,006	0,185	0,149	0,155	0,621	0,354	0,429	1,632	0,651	0,933
9	-0,521	-0,441	-0,468	-0,443	-0,394	-0,434	-0,398	-0,364	-0,407	-0,364
	0,000	0,029	0,021	0,029	0,066	0,044	0,063	0,124	0,081	0,121
10	-0,606	-0,539	-0,561	-0,538	-0,509	-0,531	-0,507	-0,486	-0,506	-0,485
	0,000	0,030	0,021	0,026	0,063	0,047	0,059	0,113	0,093	0,111
11	-1,752	-1,411	-1,447	-1,393	-0,824	-0,929	-0,762	-0,379	-0,502	-0,291
	0,005	0,457	0,413	0,487	1,442	1,155	1,633	4,198	2,930	5,797
12	-0,886	-0,657	-0,69	-0,657	-0,459	-0,515	-0,459	-0,355	-0,393	-0,352
	0,068	0,679	0,603	0,706	1,721	1,386	1,802	2,868	2,415	2,955
13	-0,721	-0,69	-0,698	-0,691	-0,665	-0,678	-0,666	-0,636	-0,651	-0,628
	0,003	0,119	0,120	0,126	0,159	0,147	0,167	0,233	0,207	0,242
14	-0,553	-0,495	-0,51	-0,493	-0,471	-0,481	-0,469	-0,46	-0,467	-0,46
	0,002	0,016	0,014	0,018	0,028	0,027	0,030	0,050	0,047	0,046
15	-0,709	-0,664	-0,672	-0,663	-0,635	-0,648	-0,632	-0,597	-0,619	-0,596
	0,000	0,051	0,046	0,057	0,080	0,066	0,090	0,141	0,110	0,138
16	-0,436	-0,385	-0,394	-0,386	-0,361	-0,371	-0,363	-0,334	-0,35	-0,339
	0,000	0,073	0,056	0,070	0,114	0,084	0,107	0,195	0,140	0,177
17	-0,748	-0,723	-0,724	-0,723	-0,691	-0,699	-0,694	-0,629	-0,654	-0,622
	0,000	0,015	0,014	0,015	0,074	0,056	0,072	0,254	0,173	0,325

Table 2: The power of fitness (bigger font) and degrees of overfitting (smaller font) obtained in the first 17 data sets.

Id		$k=3$			$k=4$			$k=5$		
	CL	FG	PFT	SFT	FG	PFT	SFT	FG	PFT	SFT
18	-0,632	-0,566	-0,593	-0,568	-0,531	-0,565	-0,531	-0,489	-0,527	-0,486
	0,003	0,042	0,029	0,035	0,139	0,092	0,168	0,401	0,214	0,442
19	-0,635	-0,579	-0,598	-0,58	-0,544	-0,56	-0,543	-0,508	-0,532	-0,511
	0,003	0,031	0,022	0,036	0,097	0,062	0,125	0,295	0,128	0,290
20	-0,677	-0,616	-0,635	-0,614	-0,592	-0,61	-0,586	-0,559	-0,58	-0,554
	0,001	0,023	0,017	0,023	0,064	0,043	0,072	0,168	0,090	0,206
21	-0,6	-0,528	-0,528	-0,528	-0,508	-0,513	-0,508	-0,483	-0,492	-0,483
	0,008	0,074	0,074	0,074	0,114	0,105	0,122	0,190	0,167	0,195
22	-0,489	-0,452	-0,466	-0,455	-0,44	-0,451	-0,445	-0,429	-0,442	-0,437
	0,002	0,007	0,009	0,009	0,011	0,009	0,011	0,021	0,016	0,016
23	-0,673	-0,599	-0,61	-0,57	-0,516	-0,563	-0,503	-0,468	-0,525	-0,443
	0,007	0,063	0,061	0,077	0,167	0,115	0,197	0,385	0,198	0,481
24	-0,701	-0,5	-0,555	-0,485	-0,341	-0,449	-0,302	-0,222	-0,352	-0,18
	0,141	1,286	1,114	1,491	2,604	1,686	3,252	4,874	2,577	6,544
25	-0,511	-0,333	-0,376	-0,333	-0,22	-0,286	-0,22	-0,169	-0,224	-0,17
	0,276	1,832	1,351	1,697	3,955	2,570	3,932	6,178	4,085	6,329
26	-0,873	-0,663	-0,697	-0,655	-0,572	-0,612	-0,568	-0,533	-0,556	-0,533
	0,003	0,029	0,022	0,024	0,063	0,047	0,060	0,083	0,088	0,105
27	-0,574	-0,502	-0,511	-0,501	-0,469	-0,487	-0,474	-0,453	-0,468	-0,457
	0,000	0,012	0,010	0,012	0,023	0,018	0,021	0,038	0,034	0,037
28	-1,885	-1,852	-1,855	-1,852	-1,797	-1,802	-1,798	-1,626	-1,639	-1,63
	0,001	0,016	0,015	0,016	0,058	0,054	0,058	0,209	0,195	0,205
29	-0,733	-0,708	-0,72	-0,711	-0,679	-0,698	-0,683	-0,635	-0,663	-0,64
	0,001	0,064	0,051	0,063	0,119	0,089	0,117	0,209	0,157	0,194
30	-0,673	-0,599	-0,61	-0,57	-0,516	-0,563	-0,503	-0,468	-0,525	-0,443
	0,007	0,063	0,061	0,077	0,167	0,115	0,197	0,385	0,198	0,481
31	-0,492	-0,467	-0,47	-0,467	-0,455	-0,46	-0,455	-0,445	-0,452	-0,446
	0,000	0,006	0,004	0,006	0,011	0,009	0,011	0,016	0,013	0,013
32	-0,664	-0,626	-0,63	-0,627	-0,606	-0,614	-0,608	-0,582	-0,592	-0,584
	0,000	0,062	0,048	0,056	0,101	0,080	0,097	0,160	0,127	0,149
33	-0,922	-0,894	-0,894	-0,894	-0,882	-0,883	-0,882	-0,875	-0,876	-0,875
	0,000	0,003	0,003	0,004	0,008	0,006	0,008	0,011	0,009	0,013
Avg.	-0,738	-0,654	-0,670	**-0,651**	-0,583	-0,612	**-0,581**	-0,521	-0,557	**-0,519**
Rank		1,394	2,818	**1,303**	1,424	3,000	**1,394**	1,485	3,000	**1,364**
		2,182	**1,091**	2,485	2,424	**1,000**	2,515	2,394	**1,061**	2,545

Table 3: The power of fitness (bigger font) and degrees of overfitting (smaller font) obtained in the last 16 data sets.

ture for PFT using Algorithm is 5 almost one. The parameters of the model associated to each structure have been randomly sampled from a 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, $N \in \{30, 100, 300, 1000, 3000, 10000, 30000\}$. The generated test sets for computing the power of generalization and degree of overfitting are of size $M = 10000$.

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` $\{u, v\} \subset C\}$

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 to obtain, on average, probability distributions with similar power of fitting (and generalization) for the different values of k and n. Thus, the scale of the obtained power of fitting (and generalization) are comparable for different settings.

- 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 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.

The obtained results are summarized in Figure 10. The figure shows a subset of the obtained results ($n \in \{80, 320, 1280\}$), 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 fitness 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. For small values of N (30 or 100), PFT has shown the best power of generalization values, especially with $k = 3$. The main reason is that, in the case of $k = 3$, the chain caveat has a lower effect over PFT.

- As k increases, the differences among the models increase, while the ranking among them is maintained. Besides, the results of CL becomes worse, and the benefits of considering the use of SFT, FG or PFT approaches becomes higher.

- As n increases, SFT, FG and PFT approaches tend to obtain worse results. 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 overfitting degrees obtained for different settings of N, n and k are summarized in Table 4. PFT obtains the lowest overfitting degrees for any configuration of N, n and k. The difference is especially remarkable for $k = 5$ and $N = 100$. SFT and FG obtain similar results. As n and k increase, the size of the training set required to obtain a low overfitting degree increases. However, it must be highlighted that FG, PFT and SFT obtain overfitting degrees of the same order of magnitude as CL.

- Execution times for different settings of k, n and N are summarized in Table 5. These values have been obtained with a non-parallel implementation of PFT and SFT. PFT is the most efficient algorithm, requiring execution times very close to the CL algorithm. As k increases, the execution time slightly increases for PFT. SFT is slower than FG in most of the configurations of

			$k=3$			$k=4$			$k=5$		
		CL	Greedy	PFT	SFT	Greedy	PFT	SFT	Greedy	PFT	SFT
80	100	0,113	0,181	0,162	0,184	0,261	0,220	0,276	0,394	0,320	0,439
80	1000	0,008	0,014	0,014	0,011	0,021	0,020	0,020	0,037	0,033	0,035
80	10000	0,001	0,001	0,001	0,001	0,002	0,002	0,002	0,003	0,003	0,003
320	100	0,166	0,238	0,215	0,249	0,320	0,272	0,349	0,462	0,381	0,523
320	1000	0,011	0,020	0,016	0,019	0,028	0,025	0,025	0,043	0,039	0,043
320	10000	0,001	0,002	0,002	0,001	0,002	0,002	0,002	0,004	0,004	0,004
1280	100	0,222	0,294	0,267	0,311	0,374	0,324	0,416	0,516	0,430	0,594
1280	1000	0,014	0,024	0,024	0,021	0,032	0,030	0,030	0,048	0,043	0,048
1280	10000	0,001	0,002	0,002	0,002	0,003	0,003	0,003	0,004	0,004	0,004

Table 4: Summary of the overfitting degrees.

n, N and k. We have observed that, for SFT, most of the execution time is consumed by the prune-and-graft procedure.

- In additional experimentation (not included in this work), SFT without the P&G operator compared to PFT obtains similar execution times, better power of fitting and similar overfitting degrees.

8 Conclusions and future work

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, see Definition 4 and Theorem 2).

In this work we have presented the family of the fractal tree algorithms. These algorithms are focused on learning MkDGs for dealing with Problem 1 and they have a computational complexity of $\mathcal{O}(k \cdot n^2 \log n)$. 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 and Liu's algorithm (see Algorithm 1). 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 directly inserted after each growing step of the fractal tree

		$k=2$	$k=3$			$k=4$			$k=5$		
n	N	CL	FG	PFT	SFT	FG	PFT	SFT	FG	PFT	SFT
80	100	0,08	0,20	0,12	0,26	0,27	0,14	0,43	0,38	0,16	0,67
320	100	2,07	7,07	2,75	6,36	7,48	2,88	12,3	8,81	3,19	20.26
1280	100	71,6	859	83,3	244	829	84,5	486	814	89,2	832
80	1000	0,41	1,11	0,80	1,35	1,58	0,83	2,25	1,97	0,87	3,12
320	1000	8,91	23,0	15,8	30,3	26,0	14,4	46,1	27,5	14,1	70,8
1280	1000	190	1251	319	592	1109	273	930	1165	262	1591
80	10000	8,88	20,6	16,0	16.6	27,2	16,4	26.4	32,6	16,6	37,9
320	10000	182	336	287	413	364	256	575	362	244	859
1280	10000	3280	6109	4864	6377	6118	4764	8559	5909	4047	15025

Table 5: Summary of the execution times (in seconds) obtained. PFT is the most efficient approach in all the configurations. FG expends more computational resources in the management of the separators than both PFT and SFT. However, SFT computes more mutual information quantities than FG and PFT. FG is more efficient than SFT in most of configurations of n, N and k mainly due to the P&G procedure. In additional experimentation, SFT without the P&G procedure is more efficient than FG and obtains execution times very close to PFT.

algorithms. In the experimental section we have measured the power of fitness and the overfitting degree in artificial and real domains. Sequential fractal tree with the prune-and-graft procedure has shown the best behavior in terms of the power of fitting, while PFT has obtained the smallest overfitting degrees (see Section 7). The fractal tree family of algorithms has shown a competitive behavior for dealing with Problem 1 and due to their efficiency are especially recommendable to deal with high dimensional domains.

In the future, we will develop massive prune-and-graft procedures for the fractal tree family of algorithms based on the features of MkDGs. The intuition consists of cutting the decomposable structures into different parts and gluing them together, increasing the likelihood of the original structure. Additionally, novel efficient learning strategies based on the separators of decomposable graphs will be developed.

The structural learning guided by the likelihood score tends to include some spurious edges which increase the probability of suffering the overfitting phenomena. We will study two strategies to alleviate this problem. First, we will adapt the presented algorithms using penalized decomposable scores in order to avoid the addition of the spurious edges. Besides, qualitative approaches based on the detection of conditional independences by means of hypothesis testing procedures will be developed. Secondly, we will study our learning algorithms in combination with methods that remove unnecessary edges such as the approach recently proposed in [18].

Finally, due to the efficiency of the proposed approaches, we will search for techniques and methods which require the use of fast algorithms for learning decomposable models. For example we will extend the work by [14] for learning mixtures of MkDGs.

References

[1] F. R. Bach and M. I. Jordan. Thin junction trees. In *Advances in Neural Information Processing Systems (NIPS)*, pages 569–576. MIT press, 2001.

[2] A. Chechetka and C. Guestrin. Efficient principled learning of thin junction trees. In *Advances in Neural Information Processing Systems (NIPS)*. Mit press, 2008.

[3] C.K. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467, 1968.

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

[5] A. Desphande, M. Garofalakis, and M. I. Jordan. Efficient stepwise selection in decomposable models. In *Uncertainty and Artificial Intelligence*, pages 128–135, 2001.

[6] G. Elidan and S. Gould. Learning bounded treewidth bayesian networks. *Journal of Machine Learning Research*, 9:2699–2731, 2008.

[7] S. Højsgaard, D. Edwards, and S. Lauritzen. *Graphical Models with R*. Springer, 2012.

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

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

[10] S. L. Lauritzen. *Graphical Models*. Oxford University Press, New York, 1996.

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

[12] L. Lu, A. Mohr, and L. Székely. Quest for negative dependency graphs. In *Recent Advances in Harmonic Analysis and Applications. In Honor of Konstantin Oskolkov.*, pages 243–258. Springer, 2013.

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

[14] M. Meila and M. I. Jordan. Learning with mixtures of trees. *Journal of Machine Learning Research*, 1:1–48, 2001.

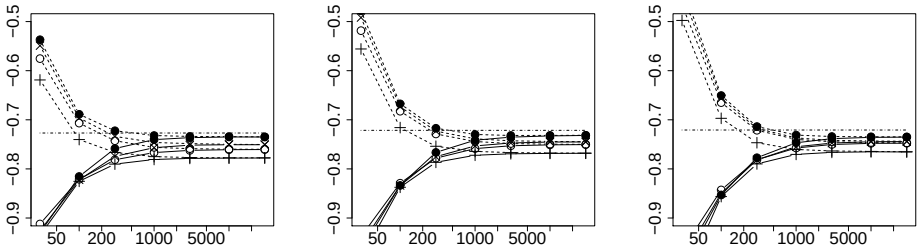
[15] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, California, 1988.

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

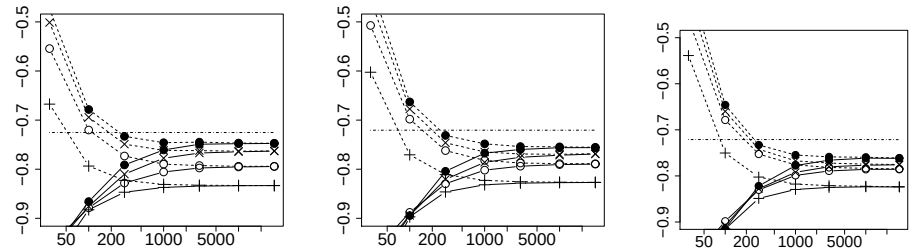
[17] N. Srebro. Maximum likelihood bounded tree-width markov networks. *Artificial Intelligence*, 143:123–138, 2003.

[18] D. Vats and R. N. Robert. A junction tree framework for undirected graphical model selection. *Journal of Machine Learning Research*, 15:147–191, 2014.

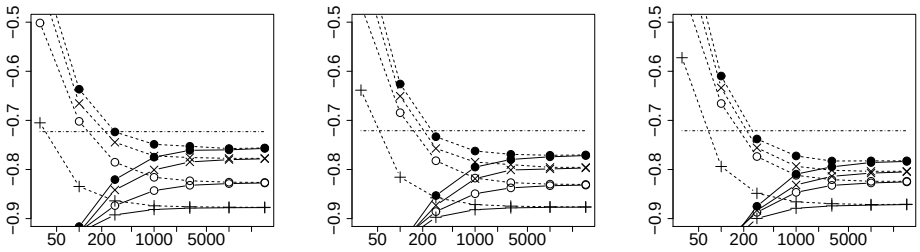
[19] T.R. Willemain and V.B. Mihoko. The distribution of node degree in maximum spanning trees. *Journal of Statistical Computation and Simulation*, 72(2):101–106, 2002.



(a) $n = 80$ and $k = 3$ (b) $n = 320$ and $k = 3$ (c) $n = 1280$ and $k = 3$

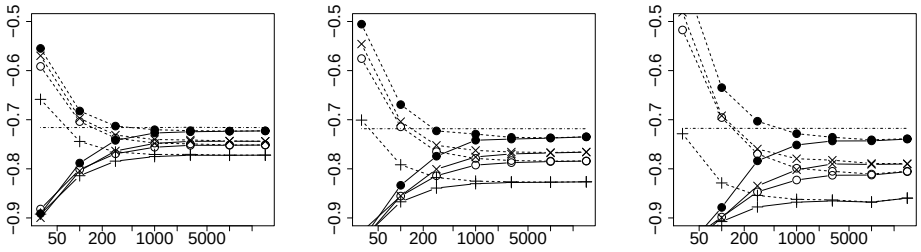


(d) $n = 80$ and $k = 4$ (e) $n = 320$ and $k = 4$ (f) $n = 1260$ and $k = 4$



(g) $n = 80$ and $k = 5$ (h) $n = 320$ and $k = 5$ (i) $n = 1260$ and $k = 5$

Figure 10: The evolution of the power of fitting (solid lines) and power of generalization (dashed lines) with respect to the number of samples in the training set. The horizontal vertical dotted line represents the power of generalization of the true model. The results obtained by the different algorithms are represented by the plus symbol for CL (+), the cross symbol for FG ($\times$), the white circle for PFT ($\circ$) and the black circle for SFT ($\bullet$).



(a) $n = 40$ and $k = 3$ (b) $n = 40$ and $k = 4$ (c) $n = 40$ and $k = 5$

Figure 11: The evolution of the power of fitting (solid lines) and power of generalization (dashed lines) with respect to the number of samples in the training set. The experiments with BTBN have been limited to $n = 40$ due to its high computational requirements. The horizontal vertical dotted line represents the power of generalization of the true model. The results obtained by the different algorithms are represented by the plus symbol for CL (+), the cross symbol for BTBN ($\times$), the white circle for PFT ($\circ$) and the black circle for SFT ($\bullet$). BTBN has obtained worse results than SFT and slightly better power of fitness than PFT, at the expense of very high execution times, e.g. for $N = 100$ and $k = 3, 4$ and 5 it requires 193, 987 and 8187 seconds, respectively. When the number of training samples is small, PFT obtained better power of generalization values than BTBN.