# Structural Learning of Simple Staged Trees

Manuele Leonelli[1*†] and Gherardo Varando[2†]

[1*]School of Science and Technology, IE University, Madrid, Spain.
[2]Image Processing Laboratory, Universitat de València, València, Spain .

*Corresponding author(s). E-mail(s): manuele.leonelli@ie.edu;
Contributing authors: gherardo.varando@uv.es;
†These authors contributed equally to this work.

**Abstract**

Bayesian networks faithfully represent the symmetric conditional independences existing between the components of a random vector. Staged trees are an extension of Bayesian networks for categorical random vectors whose graph represents non-symmetric conditional independences via vertex coloring. However, since they are based on a tree representation of the sample space, the underlying graph becomes cluttered and difficult to visualize as the number of variables increases. Here, we introduce the first structural learning algorithms for the class of simple staged trees, entertaining a compact coalescence of the underlying tree from which non-symmetric independences can be easily read. We show that data-learned simple staged trees often outperform Bayesian networks in model fit and illustrate how the coalesced graph is used to identify non-symmetric conditional independences.

**Keywords:** Asymmetric graphical models, Bayesian networks, Context-specific independence, Staged trees, Structural Learning

## 1 Introduction

Bayesian networks (BNs) are the most used statistical graphical model, providing an intuitive as well as efficient representation of multivariate data. Although the structure of the network can be expert-elicited, it is often learned from data using search algorithms that explore different network structures

(e.g. Daly, Shen, & Aitken, 2011; Neapolitan, 2004, for a review). Novel algorithms continue to appear to account for more flexible structures and to improve speed when thousands of variables are investigated (e.g. Tsagris, 2020; Wang, Gao, Yang, Tan, & Chen, 2020; Yang, Gao, Guo, & Chen, 2019).

One drawback of BNs is that they can only explicitly represent standard, symmetric conditional independence statements. However, in practice, conditional independence may faithfully hold only for specific instantiations of the conditioning variables (Shen, Choi, & Darwiche, 2020). Such independences are usually referred to as context-specific conditional independences or, more generally, as partial or local conditional independences (Pensar, Nyman, Lintusaari, & Corander, 2016).

For this reason, extensions of BNs which formally account for non-symmetric conditional independences have been proposed (Boutilier, Friedman, Goldszmidt, & Koller, 1996; Cano, Gómez-Olmedo, Moral, Pérez-Ariza, & Salmerón, 2012; Chickering, Heckerman, & Meek, 1997; DesJardins, Rathod, & Getoor, 2008; Friedman & Goldszmidt, 1996; Geiger & Heckerman, 1996; Hyttinen, Pensar, Kontinen, & Corander, 2018; Jaeger, Nielsen, & Silander, 2006; Pensar, Nyman, Koski, & Corander, 2015; Pensar et al., 2016; Poole & Zhang, 2003; Salmerón, Cano, & Moral, 2000) and there has been an increasing interest in formalizing the notion of non-symmetric independence (Corander, Hyttinen, Kontinen, Pensar, & Väänänen, 2019; Nicolussi & Cazzaro, 2021; Shen et al., 2020; Tikka, Hyttinen, & Karvanen, 2019). With the exception of Jaeger et al. (2006) and Pensar et al. (2015), all the above-cited models somehow lose the intuitiveness of BNs since they cannot represent all the models' information in a unique graph.

Staged trees (Collazo, Görgen, & Smith, 2018; Smith & Anderson, 2008) are a flexible class of statistical graphical models for categorical variables which, starting from an event tree, explicitly and graphically represent any non-symmetric conditional independence by a partitioning of the vertices of the graph. Staged trees have recently gained popularity thanks to the deployment of the `stagedtree` R package (Carli, Leonelli, Riccomagno, & Varando, 2022) and to their use for causal reasoning at Google's Deepmind (Genewein et al., 2020).

Despite being able to represent any non-symmetric independences in a unique graph, the visualization of staged trees becomes cluttered and difficult to interpret as the dimensionality of the problem increases. For this reason, approaches to compress the large amount of information about the complex dependence structure encoded by the staged tree have been introduced. Varando, Carli, and Leonelli (2021) defined algorithms to transform a staged tree into a labeled DAG informing about the type of dependence existing between pairs of variables. Smith and Anderson (2008) devised a coalescence of the vertices of a staged tree, giving a more compact graphical representation called chain event graph (CEG), equivalent to the staged tree.

Here, we demonstrate that for generic staged trees learned from data, the coalescence rule of Smith and Anderson (2008) merges only a small number of

vertices, and thus, the practical gain of the CEG representation is often very limited. The class of *simple* staged trees (Smith & Anderson, 2008) is one such that the coalescence process into a CEG is optimal, in a sense that we formalize below. The focus of this paper is on this specific subclass of staged trees.

Although there are now many different structural learning algorithms for staged trees and CEGs (Barclay, Hutton, & Smith, 2013; Carli et al., 2022; Carli, Leonelli, & Varando, 2023; Collazo & Smith, 2016; Cowell & Smith, 2014; Freeman & Smith, 2011; Leonelli & Varando, 2022, 2023b, 2024; Silander & Leong, 2013), to date, there are no methods that learn simple staged trees from data. Here, after formalizing the relationship between simple staged trees and BNs, we introduce novel learning algorithms for simple staged trees and illustrate their use in a variety of datasets. Our applications demonstrate that simple staged trees, while still often outperforming BNs in terms of model fit and predictive accuracy, provide a concise graphical representation of a wide array of non-symmetric conditional independences. Furthermore, we provide a first discussion of the equivalence class of a simple staged tree. In order to assess causal relationships, it is fundamental to explore the equivalence class of a model, and there has been a growing interest in developing algorithms to perform this task (e.g. Görgen, Bigatti, Riccomagno, & Smith, 2018; Görgen & Smith, 2018; Leonelli & Varando, 2023a).

## 2 Simple Bayesian Networks

Before considering staged trees, we need to introduce BNs and a novel class, termed *simple*, for which the underlying graph is constrained. Let $G = ([p], F)$ be a directed acyclic graph (DAG) with vertex set $[p] = \{1, \ldots, p\}$ and edge set $F$. Let $\boldsymbol{X} = (X_i)_{i\in[p]}$ be categorical random variables with joint mass function $P$ and sample space $\mathbb{X} = \times_{i\in[p]}\mathbb{X}_i$. For $A \subset [p]$, we let $\boldsymbol{X}_A = (X_i)_{i\in A}$ and $\boldsymbol{x}_A = (x_i)_{i\in A}$ where $\boldsymbol{x}_A \in \mathbb{X}_A = \times_{i\in A}\mathbb{X}_i$. We say that $P$ is Markov to $G$ if, for $\boldsymbol{x} \in \mathbb{X}$,

$$P(\boldsymbol{x}) = \prod_{k\in[p]} P(x_k \mid \boldsymbol{x}_{\Pi_k}), \tag{1}$$

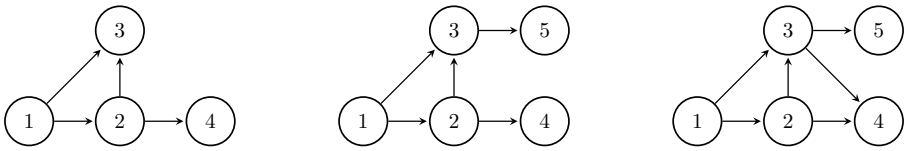
where $\Pi_k$ is the parent set of $k$ in $G$ and $P(x_k|\boldsymbol{x}_{\Pi_k})$ is a shorthand for $P(X_k = x_k|\boldsymbol{X}_{\Pi_k} = \boldsymbol{x}_{\Pi_k})$.

It is customary to label the vertices of a BN so as to respect the topological order of $G$, i.e. a linear ordering of $[p]$ for which only pairs $(i, j)$ where $i$ appears before $j$ in the order can be in the edge set. Of course, there can be multiple permutations of $[p]$ which respect the topological order. Henceforth, we assume that $1, 2, \ldots, p$ is a topological order of $G$.

The ordered Markov condition implies conditional independences of the form

$$X_i \perp\!\!\!\perp \boldsymbol{X}_{[i-1]\setminus\Pi_i} \mid \boldsymbol{X}_{\Pi_i}. \tag{2}$$

Henceforth, $P$ is assumed to be strictly positive.




**Definition 1** Let $G$ be a DAG and $P$ Markov to $G$. The *Bayesian network* model (associated to $G$) is

$$\mathcal{M}_G = \{P \in \Delta^{\circ}_{|\mathbb{X}|-1} \,|\, P \text{ is Markov to } G\}.$$

where $\Delta^{\circ}_{|\mathbb{X}|-1}$ is the $(|\mathbb{X}|-1)$-dimensional open probability simplex.

In practice, it is often convenient to work with BNs whose DAG is *decomposable*. A DAG $G = ([p], F)$ is said to be decomposable if, for every $i \in [p]$, the set $\Pi_i$ is such that there exists a $j \in \Pi_i$ for which $\Pi_i = \Pi_j \cup \{j\}$. For decomposable BNs, probabilistic inference can be performed exactly using fast algorithms, so that often non-decomposable BNs are first transformed into decomposable ones via the moralization process (Darwiche, 2009). Decomposable BN models are regular exponential families, different from generic BNs, which are curved (Geiger, Heckerman, King, & Meek, 2001).

Here, we consider a novel class of DAGs.

**Definition 2** Let $G$ be a DAG and $\pi$ a topological order of $G$. We say that $G$ is *simple* with respect to $\pi$ if for all $i \in [p-1]$, $\Pi_{\pi(i+1)} \subseteq \Pi_{\pi(i)} \cup \{\pi(i)\}$. We say that $G$ is simple if there exists a topological order $\pi$ of $G$ such that $G$ is simple with respect to $\pi$.

The DAG associated with a total independence model, i.e. having an empty edge set, is simple for any permutation of the labeling of the vertices. Similarly, the DAG associated with the saturated model, i.e. there is an edge $(i, j)$ or $(j, i)$ for any two $i, j \in [p]$, is also simple for the only topological order associated to such a DAG. Figure 2 gives further illustrations of the concept of a simple DAG. The DAG in Figure 2 (left) has two topological orders (1234 and 1243) and it is simple. The DAG in Figure 2 (center) has three topological orders (12345, 12354, and 12435) but for none of these the condition in Definition 2 is met, thus it is not simple.

The following result is a straightforward consequence of Definition 2.

**Proposition 1** *Every simple DAG is decomposable.*

The converse is not true, as exemplified by the center DAG in Figure 2, which is decomposable but not simple. From Proposition 1 it is clear that simple BN models are regular exponential families.

Let $1, \ldots, p$ be a topological order and fix this order. Any DAG $G = ([p], F)$ can be transformed into a simple DAG $G' = ([p], F')$ by letting the parent

sets in $G'$ $\Pi'_{i-1} = \Pi_{i-1} \cup \{\Pi_i \setminus \Pi_{i-1}\}$. So, just as in moralization of DAGs, *simplification* transforms the original DAG by adding edges. For the DAG in Figure 2 (middle), the simplification process transforms it into the DAG in Figure 2 (right) by the addition of the edge $(3, 4)$.

We refer to Section 3.4 for a discussion on how to practically interpret the assumption of simplicity.

# 3 Non-Symmetric Models Based on Trees

Our focus is on models created from trees, different from BNs which are based on DAGs.

## 3.1 X-Compatible Staged Trees

Consider a $p$-dimensional random vector $\boldsymbol{X}$ taking values in the product sample space $\mathbb{X}$. Let $(V, E)$ be a directed, finite, rooted tree with vertex set $V$, root node $v_0$ and edge set $E$. For each $v \in V$, let $E(v) = \{(v, w) \in E\}$ be the set of edges emanating from $v$ and $\mathcal{C}$ be a set of labels.

**Definition 3** An **X**-compatible staged tree is a triple $T = (V, E, \theta)$, where $(V, E)$ is a rooted directed tree and:

1. $V = v_0 \cup \bigcup_{i\in[p]} \mathbb{X}_{[i]}$;
2. For all $v, w \in V$, $(v, w) \in E$ if and only if $w = \boldsymbol{x}_{[i]} \in \mathbb{X}_{[i]}$ and $v = \boldsymbol{x}_{[i-1]}$, or $v = v_0$ and $w = x_1$ for some $x_1 \in \mathbb{X}_1$;
3. $\theta : E \to \mathcal{L} = \mathcal{C} \times \cup_{i\in[p]}\mathbb{X}_i$ is a labelling of the edges such that $\theta(v, \boldsymbol{x}_{[i]}) = (\kappa(v), x_i)$ for some function $\kappa : V \to \mathcal{C}$. The function $k$ is called the colouring of the staged tree $T$.

If $\theta(E(v)) = \theta(E(w))$ then $v$ and $w$ are said to be in the same *stage*.

Therefore, the equivalence classes induced by $\theta(E(v))$ form a partition of the internal vertices of the tree in *stages*.

Definition 3 first constructs a rooted tree where each root-to-leaf path, or equivalently each leaf, is associated with an element of the sample space $\mathbb{X}$. Then, a labeling of the edges of such a tree is defined where labels are pairs with one element from a set $\mathcal{C}$ and the other from the sample space $\mathbb{X}_i$ of the corresponding variable $X_i$ in the tree. By construction, **X**-compatible staged trees are such that two vertices can be in the same stage if and only if they correspond to the same sample space. Although staged trees can be more generally defined without imposing this condition (see e.g. Collazo et al., 2018), henceforth, and as common in practice, we focus on **X**-compatible staged trees only (see Leonelli, 2019, for an example of a non **X**-compatible tree).

Figure 1 reports an $(X_1, X_2, X_3)$-compatible staged tree over three binary variables. The *coloring* given by the function $\kappa$ is shown in the vertices and each edge $(\cdot, (x_1, \ldots, x_i))$ is labeled with $X_i = x_i$. The edge labeling $\theta$ can be

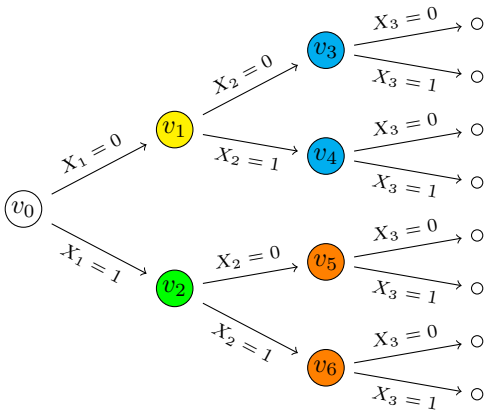



**Fig. 1** An example of an **X**-compatible staged tree.

read from the graph by combining the text label and the color of the emanating vertex. The staging of the staged tree in Figure 1 is given by the partition $\{v_0\}$, $\{v_1\}$, $\{v_2\}$, $\{v_3, v_4\}$ and $\{v_5, v_6\}$.

The parameter space associated to an **X**-compatible staged tree $T = (V, E, \theta)$ with labeling $\theta : E \to \mathcal{L}$ is defined as

$$\Theta_T = \Big\{ \boldsymbol{y} \in \mathbb{R}^{|\theta(E)|} \mid \forall\, e \in E, y_{\theta(e)} \in (0,1) \text{ and } \sum_{e \in E(v)} y_{\theta(e)} = 1 \Big\}. \tag{3}$$

Equation (3) defines a class of probability mass functions over the edges emanating from any internal vertex coinciding with conditional distributions $P(x_i|\boldsymbol{x}_{[i-1]})$, $\boldsymbol{x} \in \mathbb{X}$ and $i \in [p]$. In the staged tree in Figure 1 the staging $\{v_3, v_4\}$ implies that the conditional distribution of $X_3$ given $X_1 = 0$ and $X_2 = 0$, represented by the edges emanating from $v_3$, is equal to the conditional distribution of $X_3$ given $X_1 = 0$ and $X_2 = 1$. A similar interpretation holds for the staging $\{v_5, v_6\}$. This, in turn, implies that $X_3 \perp\!\!\!\perp X_2 | X_1$, thus illustrating that the staging of a tree is associated with conditional independence statements. On the other, the fact that $v_1$ and $v_2$ are each in their individual stage means that there is no independence between $X_1$ and $X_2$.

Let $\boldsymbol{l}_T$ denote the leaves of a staged tree $T$. Given a vertex $v \in V$, there is a unique path in $T$ from the root $v_0$ to $v$, denoted as $\lambda(v)$. The *depth* of a vertex $v \in V$ equals the number of edges in $\lambda(v)$. For any path $\lambda$ in $T$, let $N(\lambda) = \{e \in E : e \in \lambda\}$ denote the set of edges in the path $\lambda$.

**Definition 4** The *staged tree model* $\mathcal{M}_T$ associated to the **X**-compatible staged tree $(V, E, \theta)$ is the image of the map

$$\begin{aligned} \phi_T : \Theta_T &\to \Delta^{\circ}_{|\boldsymbol{l}_T|-1} \\ \boldsymbol{y} &\mapsto \Big( \textstyle\prod_{e \in N(\lambda(l))} y_{\theta(e)} \Big)_{l \in \boldsymbol{l}_T} \end{aligned} \tag{4}$$

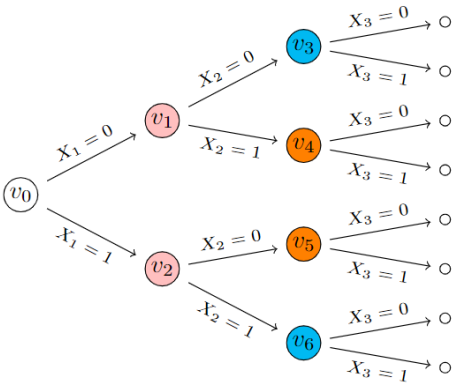



**Fig. 2** An example of a staged tree not associated to any BN $G$.

Therefore, staged trees models are such that atomic probabilities are equal to the product of the edge labels in root-to-leaf paths and coincide with the usual factorization of mass functions via recursive conditioning.

## 3.2 Staged Trees and Bayesian Networks

Although the relationship between BNs and staged trees was already formalized by Smith and Anderson (2008), a formal procedure to represent a BN as a staged tree has been recently introduced in Duarte and Solus (2023).

Assume $\boldsymbol{X}$ is topologically ordered with respect to a DAG $G$ and consider an $\mathbf{X}$-compatible staged tree with vertex set $V$, edge set $E$ and labeling $\theta$ defined via the coloring $\kappa(\boldsymbol{x}_{[i]}) = \boldsymbol{x}_{\Pi_i}$ of the vertices. The staged tree $T_G$, with vertex set $V$, edge set $E$ and labeling $\theta$ so constructed, is called *the staged tree model of* $G$. Importantly, $\mathcal{M}_G = \mathcal{M}_{T_G}$, i.e. the two models are exactly the same since they entail exactly the same factorization of the joint probability. Clearly, the staging of $T_G$ represents the Markov conditions associated with the graph $G$.

For instance, the staged tree in Figure 1 can be constructed as the $T_G$ from the BN with DAG $X_2 \leftarrow X_1 \rightarrow X_3$. Conversely, consider the staged tree in Figure 2. The pink staging of $v_1$ and $v_2$ implies that $X_1 \perp\!\!\!\perp X_2$, which can be represented by a DAG. However, the blue staging implies that the conditional distribution of $X_3$ given $X_2 = X_1 = 0$ is equal to the conditional distribution of $X_3$ given $X_2 = X_1 = 1$. Such a constraint cannot be explicitly represented by the DAG of a BN, and therefore there is no DAG $G$ such that $\mathcal{M}_G = \mathcal{M}_{T_G}$, i.e. there is no BN which is equivalent to the staged tree in Figure 2.

More generally, Smith and Anderson (2008) demonstrated that any BN can be represented as an equivalent staged tree, while the converse is not true (as in Figure 2).

## 3.3 Chain Event Graphs

Although extremely rich since they can represent any non-symmetric conditional independence (see e.g. Figure 2), the class of staged tree models becomes difficult to visualize as the size of the tree increases. Smith and Anderson (2008) devised a transformation of a staged tree which, while not changing the statistical model, reduces (in theory) the number of vertices and edges of the graph.

Let $T_v$ be the subtree of the $(X_1, \ldots, X_p)$-compatible staged tree $T$ rooted at $v \in V$ and suppose that $v$ has depth $k$. Then $T_v$ is a $(X_{k+1}, \ldots, X_p)$-compatible staged tree with labeling induced by $\theta$. Two internal vertices $v, w \in V$ are said to be in the same *position* if $T_v = T_w$. Of course, two vertices can be in the same position only if they are in the same stage. Positions give a coarser partition of the vertex set than stages, i.e. the number of positions is bigger or equal to the number of stages. All leaves of the staged tree are trivially merged in the same position, denoted $w_\infty$.

**Definition 5** Let $T = (V, E, \theta)$ be an $\mathbf{X}$-compatible staged tree and $w_1, \ldots, w_n, w_\infty$ its positions. The associated *chain event graph* $C_T = (V_C, E_C, \kappa_C)$ is a multi-edge graph $(V_C, E_C)$ together with a coloring of the vertices $\kappa_C$, such that $V_C = \{w_1, \ldots, w_n, w_\infty\}$ and the edges $E_C$ are constructed as follows: for every edge $(v, v') \in E$ let $(w, w', x) \in E_C$ be a labelled edge in the CEG, where $w$ and $w'$ are the positions of $v$ and $v'$ and $\theta(v, v') = (\kappa(v), x)$. The coloring $\kappa_C$ over $V_C$ is simply the one induced by the coloring $\kappa$ defining $\theta$.

In a CEG all vertices of the staged tree in the same position are merged in a unique vertex. Furthermore, pairs of edges that connect vertices in the same positions are also merged. Therefore, the number of vertices and edges of the CEG is smaller than that of the equivalent staged tree.

The construction of a CEG is illustrated for the staged tree in Figure 3. The positions of this tree are $w_0 = \{v_0\}$, $w_1 = \{v_1\}$, $w_2 = \{v_2\}$, $w_3 = \{v_3, v_5\}$, $w_4 = \{v_4\}$, $w_5 = \{v_6\}$, $w_6 = \{v_7, v_9, v_{10}, v_{11}\}$ and $w_7 = \{v_8, v_{12}, v_{13}, v_{14}\}$. For instance, $v_1$ and $v_2$, although in the same stage, are not in the same position since the subtrees rooted at $v_1$ and $v_2$ are not identical (as an example $v_4$ and $v_6$ are not in the same stage). These positions, together with the leaf $w_\infty$, are the vertices of the equivalent CEG reported in Figure 4. The vertices $v_3$ and $v_5$ are in the same position $w_3$ of the staged tree. Their edge labeled $X_3 = 0$ leads to $v_7$ and $v_{11}$, respectively, which are in the same position $w_6$. Therefore, the two edges are merged in the CEG as can be noticed in Figure 4. Even in this simple example, it is apparent that the CEG is more compact than the associated staged tree, with a reduction from fifteen to eight internal vertices.

It is possible to define the CEG model $\mathcal{M}_{C_T}$ by simply adapting equations (3)-(4) to the CEG. Notice that, as in a staged tree, the labeling over the edges emanating from an internal vertex defines a probability mass function, and root-to-leaf paths are associated with atomic events, whose probabilities

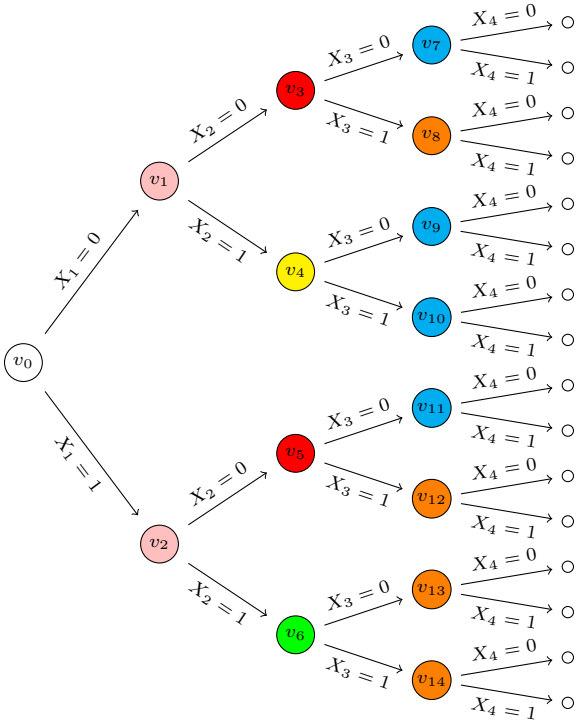



**Fig. 3** Example of an **X**-compatible staged tree over four binary random variables.

can be computed as products of the associated edge labels. Importantly, Smith and Anderson (2008) demonstrated that $\mathcal{M}_T = \mathcal{M}_{C_T}$.

The CEG does not only provide a more compact graphical representation of the model but also a framework to perform fast probability propagation (Thwaites, Smith, & Cowell, 2008) and to intuitively read conditional independences from the graph (Thwaites & Smith, 2015). Illustrations of how to read independences in the CEG are given in Section 6.

## 3.4 Simple Staged Trees

Staged trees learned from data are such that the graphical simplification of transforming the tree into a CEG is often minimal. Figure 3 gives an idea in this direction since vertices at depth one ($v_1$ and $v_2$), although in the same stage, are not in the same position. In general, vertices close to the root are less likely to be in the same position since the associated subtrees require many symmetries.

**Definition 6** An **X**-compatible staged tree $T = (V, E, \theta)$ is *simple* if the vertex partitions into stages and positions coincide. That is, if for every vertices $v, w \in V$,

$$\theta(E(v)) = \theta(E(w)) \Rightarrow T_v = T_w.$$

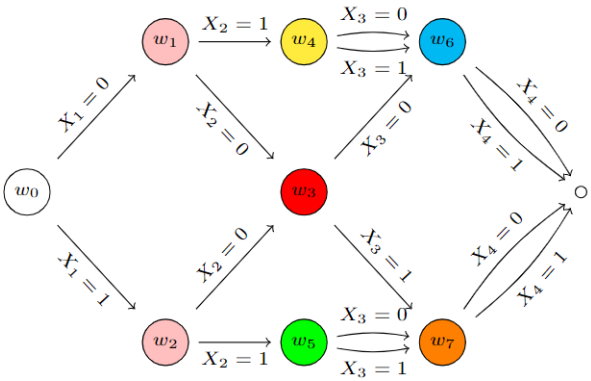



**Fig. 4** The CEG equivalent to the staged tree in Figure 3.

Simple staged trees lead to a very compact CEG since each stage corresponds to exactly one vertex in the CEG. The staged trees in Figures 1 and 2 are simple, while the staged tree in Figure 3 is not simple: indeed, its CEG has two vertices belonging to the same stage ($w_1$ and $w_2$).

Definition 6 gives a graphical condition for a staged tree to be simple, but does not clearly highlight what assumptions underlie simplicity, which is deeply connected to the order $(X_1, \ldots, X_p)$ chosen for the variables. Fix such an order and assume that two vertices associated with $X_i$ are in the same stage: that is, $P(x_i|\boldsymbol{x}_{[i-1]}) = P(x_i|\boldsymbol{x}'_{[i-1]})$, for all $x_i \in \mathbb{X}_i$ and $\boldsymbol{x}_{[i-1]}, \boldsymbol{x}'_{[i-1]} \in \mathbb{X}_{[i-1]}$. Then in a simple staged tree the previous equality implies, for all $j > i$, that

$$P(x_j|\boldsymbol{x}_{[i-1]}, \boldsymbol{x}_{[j-1]\setminus[i-1]}) = P(x_j|\boldsymbol{x}'_{[i-1]}, \boldsymbol{x}_{[j-1]\setminus[i-1]}), \tag{5}$$

for all $x_j \in \mathbb{X}_j$ and all $\boldsymbol{x}_{[j-1]\setminus[i-1]} \in \mathbb{X}_{[j-1]\setminus[i-1]}$. In words, for all variables $X_j$ following $X_i$ in the assumed order, it must be that that the conditional probability distributions of $X_j$ are equal in the contexts including $\boldsymbol{x}_{[i-1]}$ and $\boldsymbol{x}'_{i-1}$.

A simple BN, with respect to a topological order $\pi$, is such that the equalities in Equation (5) represent symmetric conditional independence. More generally, in a simple staged tree, these may represent non-symmetric independence statements.

To illustrate the assumption of simplicity, consider the staged tree in Figure 5 inspired by Christchurch Health and Development Study (Fergusson & Horwood, 2001), studying the effect of the family's social background, the economic status, and the number of family life events on a child's health, measured by rates of hospital admission. We consider three variables: $X_1$ - social and economic situation (low/avg/high); $X_2$ - number of family life events (low/avg/high); $X_3$ - hospital admission (no/yes). An $(X_1, X_2, X_3)$-compatible staged tree is reported in Figure 5. It can be seen that the vertices $v_3$ and $v_4$ are in the same stage, i.e. the distribution of family life events is the same for low and average social and economic backgrounds. Any simple staged tree

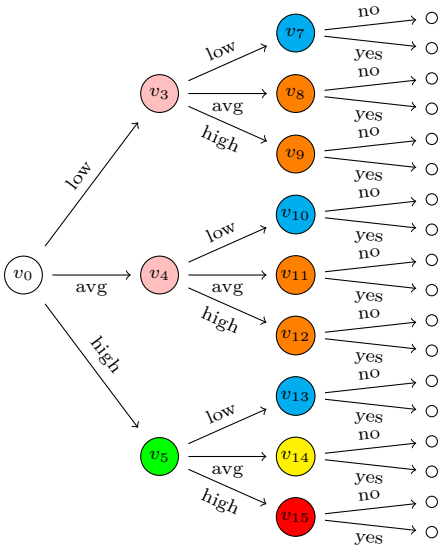



**Fig. 5** A simple staged tree for the Christchurch Health and Development Study.

with staging $\{v_3, v_4\}$ must then be such that $v_7$ and $v_{10}$, $v_8$ and $v_{11}$, and $v_9$ and $v_{12}$ are in the same stage. In words, assuming that the distribution of family life events is the same for low and average social and economic backgrounds implies, for instance, that the distribution of hospital admission is the same for $(X_1 = low, X_2 = low)$ and $(X_1 = avg, X_2 = low)$. It can be seen that the staged tree in Figure 5 is indeed simple.

The following proposition, which formalizes a result that first appeared in Smith and Anderson (2008), establishes the relationship between BNs and simple staged trees.

**Proposition 2** *Let $T_G$ be the staged tree equivalent to a BN with DAG $G$. Then $T_G$ is simple if and only if $G$ is simple.*

The result follows from Theorem 3 of Smith and Anderson (2008). So, in particular, if $T_G$ is simple, then $G$ is also decomposable, but the converse is not true. Another consequence is that if a staged tree is simple, then it is either equivalent to a simple BN or not equivalent to any BN. For example, the simple staged tree in Figure 1 is equivalent to the BN $X_2 \leftarrow X_1 \rightarrow X_3$, while the simple staged tree in Figure 2 is not equivalent to any $G$.

A result similar in spirit to Proposition 2 was derived in Duarte and Solus (2023) formalizing the equivalence between decomposable BNs and *balanced* staged trees, which are characterized by a polynomial condition based on interpolating polynomials (Görgen, Leonelli, & Smith, 2015), holding for all simple

staged trees. Görgen, Leonelli, and Marigliano (2022) demonstrated that balanced staged trees, and thus also simple ones, are regular exponential families. Therefore, the BIC is an asymptotically valid rule for model selection.

# 4 Learning and Reasoning with Simple Staged Trees

We now discuss the two main contributions of this paper: machine learning algorithms to learn simple staged trees from data, and an initial characterization of the equivalence class of a simple staged tree.

## 4.1 Structural Learning of Simple Staged Trees

We next discuss the first structural learning algorithms to learn simple staged trees from data. They can be classified into three types: (i) by learning a generic staged tree from data and then simplifying the staged tree (in a similar fashion to the simplification of BNs); (ii) by directly learning a simple staged tree from data given an ordering of the variables; (iii) directly learning a simple staged tree from data but without assuming a variable ordering a priori.

To *simplify* a staged tree, as described in Collazo et al. (2018), it is informally sufficient to use positions as stages. More precisely, given an $\mathbf{X}$-compatible staged tree $T = (V, E, \theta)$ where $\theta((v, \mathbf{x}_{[i]})) = (\kappa(v), x_i)$, let $T^s = (V, E, \theta^s)$ be the staged tree obtained by $\theta^s(v, \mathbf{x}_{[i]}) = (w(v), \mathbf{x}_i)$, where for each $v \in V$, $w(v)$ is the position of $v$ in $T$. It is easy to see that $T^s$ is a simple staged tree. The labeling $\theta^s$ is such that the resulting model $\mathcal{M}_{T^s}$ is the smallest staged tree model that includes $\mathcal{M}_T$.

Given the above, we can define a first strategy for learning a simple $\mathbf{X}$-compatible staged tree from data: (1) Use a learning algorithm to estimate an $X$-compatible staged tree $T$ (for instance, any algorithm of the `stagedtrees` R package); (2) simplify $T$ into $T^s$. Such a learning strategy has the advantage of being extremely simple to implement, and it does not add a significant computational complexity given that the position identification can be done very efficiently (Shenvi & Smith, 2020). Nevertheless, we will show that, as a drawback, it outputs CEGs with a lot of positions, rendering them intractable and not easily interpretable.

It is intuitively more efficient to learn a simple staged tree directly from data. We propose two greedy methods for this task which assume a fixed variable ordering. The first simple algorithm is a hill-climbing joining of stages, imposing, for each depth, that stages coincide with positions: (1) start from the full tree where each vertex is in a different stage and set $i = 1$. (2) run a hill-climbing joining of stages for nodes at depth $i$ to optimize the BIC score. (3) impose the stages at level $i$ to be positions by joining relevant stages in all the sub-trees (it is sufficient to join stages at depth $i+1$). Set $i = i+1$ and repeat from (2) until the whole tree is learned. We name this first method *marginal* algorithm since the BIC score is optimized for the (marginal) log-likelihood at a specific depth (i.e. for a specific variable), disregarding the effect of joining the

**Algorithm 1** Marginal Algorithm for Simple Staged Trees

**Require:** A dataset $\mathcal{D}$ over categorical variables $X_1, \ldots, X_p$
**Ensure:** A simple staged tree $T$

1: Initialize $T$ as the full staged tree with staging $U_1, \ldots, U_{p-1}$
2: $score \leftarrow BIC(T)$
3: $T^* \leftarrow T$
4: $i \leftarrow 1$
5: **while** $i \leq p-1$ **do**
6: $indicator \leftarrow 1$
7: **while** $indicator \neq 0$ **do**
8: **for** every pair of stages $u_j, u_s \in U_i$ **do**
9: Construct $T^{'}$ by merging $u_i$ and $u_j$
10: **if** $BIC(T^{'}) < BIC(T^*)$ **then**
11: $score \leftarrow BIC(T^{'})$
12: $T^* \leftarrow T^{'}$
13: **end if**
14: **end for**
15: **if** $T = T^*$ **then**
16: $indicator \leftarrow 0$
17: **else if** $T \neq T^*$ **then**
18: $T \leftarrow T^*$
19: **end if**
20: **end while**
21: **if** $i \neq p-1$ **then**
22: Simplify $T$ and $T^*$
23: **end if**
24: $i \leftarrow i+1$
25: **end while**
26: **Return:** $T$

stages for the sub-trees (the subsequent variables in the ordering). A detailed description of the steps of this procedure is given in Algorithm 1.

The second fixed variable order algorithm corresponds to a hill-climbing joining of positions to optimize the BIC score, we name this algorithm *total* in contrast with the former approach: (1) start from the full tree where each node is in a different stage and set $i = 1$. (2) Run a hill-climbing joining of positions for nodes at depth $i$ to optimize the BIC score. (3) Set $i = i + 1$ and repeat from the previous step until the whole tree is learned.

The marginal algorithm is computationally faster since the BIC score can be decomposed across the depth of the tree and thus, at each step of the search, minimal operations are needed to compute the updated score. Conversely, the total algorithm is more expensive, but the optimized score is the actual BIC of the model. The pseudo-code for the total algorithm is a simple variation of the one given in Algorithm 1.The simplification process in lines 21-23 needs to

be moved after line 9 so that it is carried out before the evaluation of the BIC score (line 10).

If the variables' ordering is not known or it cannot be assumed beforehand, it is possible to combine the previous marginal algorithm with a greedy order search. Specifically, the marginal algorithm can be coupled with the greedy approach to order search implemented in the `stagedtrees` R package (Carli et al., 2022). The greedy method works by building the tree one variable at a time. Starting from the root and adding, at each step, the variable that best improves the model score (e.g. BIC). We denote this algorithm combination *greedy marginal*. Conversely, the total algorithm cannot be coupled with the greedy approach for order search since the total optimization for stages/positions at a certain depth also affects vertices at subsequent depths. Thus, for the total method, we resort to performing an exhaustive search over the $p!$ possible variables orderings. Obviously, such an exhaustive search is feasible only for relatively small problems ($p \leq 7$ in our computational study of Section 5).

A simple implementation of all the algorithms based on the `stagedtrees` R package (Carli et al., 2022) is freely available in the repository gherardovarando/simple_stagedtrees together with the code to reproduce all the examples, simulations and applications.

## 4.2 Equivalence in Simple Staged Trees

Determining the equivalence class of a statistical graphical model, such as a BN or a staged tree, is a fundamental task to determine if relationships between variables can be assumed to have a causal meaning (Pearl, 2009; Shafer, 1996). By equivalence class of a staged tree $T$, we refer to all staged trees $T'$ whose models $\mathcal{M}_{T'} = \mathcal{M}_T$ and therefore cannot be discriminated from observational data. Algorithms that explore the equivalence class of a staged tree have been recently introduced (Görgen et al., 2018; Görgen & Smith, 2018). Such algorithms could be, of course, used to output all the staged trees equivalent to a given simple staged tree. However, they do not provide a graphical characterization of when two staged trees are equivalent, but a polynomial one. Leonelli and Varando (2023a) provided a first partial graphical criterion to assess when two staged trees are not equivalent.

Here, we provide a first discussion of the properties of the equivalence class for simple staged trees.

*Remark 1* The equivalence class of a simple staged tree $T$ does not necessarily consists of simple staged trees only.

This can be easily seen by considering a DAG $G$ such that it is simple with respect to a topological order, but not with respect to another. Let $T_G$ and $T'_G$ be the two staged trees associated to such a DAG and the two orders. Then $\mathcal{M}_{T_G} = \mathcal{M}_{T'_G}$ but one is simple and the other is not. Therefore, by using structural learning algorithms for simple staged trees, one would often learn

models that are not equivalent by considering different variable orderings. This differs from the recent result of Duarte and Solus (2021) which considers a different subclass of staged trees called CStrees. The equivalence class of a given CStree can only include other CStrees.

While Remark 1 seems a drawback at first, it is worth noticing that such a negative result introduce an asymmetry between different trees in the same statistical equivalence class and thus it could help identifying (possibly causal) orderings of the variables. In other words, by considering simple staged trees only we reduce the sets of equivalent candidates belonging to the same equivalence class.

*Remark 2* Let $T$ and $T'$ be two equivalent staged trees, i.e. $\mathcal{M}_T = \mathcal{M}_{T'}$. Then the simplified staged trees $T^s$ and $T'^s$ may not be equivalent.

To see this consider the case of two equivalent staged trees, where one of the two is simple. Then simplification of the simple tree does not change its staging, whilst the simplification of the other does. Therefore the two simplified trees are not equivalent.

# 5 Experiments

We perform computational experiments to showcase the use of the proposed simple staged tree learning algorithms: first we conduct a simulation study with generated simple staged trees as ground truth, and then we consider nine different datasets from the literature on graphical models.

## 5.1 Simulation Study

We perform a simulation study to investigate the performances of the proposed simple staged tree learning algorithms with fixed variables' order. We compare the classical backward hill-climbing algorithm (Carli et al., 2022) (BHC) and the simplified staged tree against the proposed simple staged tree learning algorithms: the marginal and total approaches described in Section 4.1.

In the experiment, we consider $\boldsymbol{X}$-compatible staged trees with six binary variables. A random simple staged tree starting from a full staged tree is constructed depending on a parameter $q$, fixing the probability that any pair of vertices at the same depth in the tree are in the same position. Thus, higher values of $q$ correspond to trees with fewer positions. In the experiment we consider values $q = 0.2, 0.5, 0.7$. We then simulate observations from the tree, considering sample sizes $N = 25, 50, 75, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1250, 1500, 1750, 2000$. For each combination of $q$ and $N$, we generate 100 different datasets from 100 different random simple staged trees. For each generated dataset, staged trees are learned with the BHC, marginal, and total algorithms. The simplified version of the tree learned with the BHC algorithm is also considered.

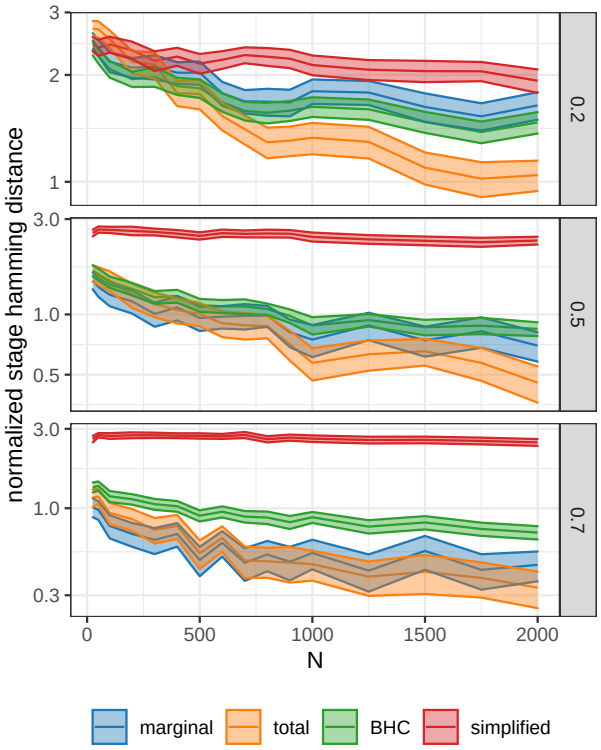



**Fig. 6** Average results and 95% confidence intervals for the normalized stage Hamming distances between estimated and true staged trees.

We compare the learned staged trees to the true generating tree by computing the *normalized Hamming stage distance*, which we define as the sum along the depth of the tree of the average number of vertices whose coloring must be changed to obtain the true generating model. We report the average and confidence intervals of the normalized Hamming stage distances in Figure 6.

We notice that, as the sample size increases, the backward hill climbing method, the marginal, and the total algorithms learn models closer to the true data-generating staged tree. Conversely, the simplified trees obtained from the BHC estimates do not converge to the true models. We thus deduce that specialized simple staged tree learning algorithms are needed to obtain simple models without sacrificing consistency. As expected, the total algorithm is to be preferred, especially for more complex models and higher sample sizes.

## 5.2 Computational Study

Next, we consider nine datasets from the literature of probabilistic graphical models to assess the performance of simple staged trees and different estimation procedures. If required, variables are discretized using the equal-frequency method. For each dataset, eight different models are considered. First, a BN

| Dataset | # Vars. | # Obs. | # Sample Space |
|---|---|---|---|
| asia | 8 | 5000 | 256 |
| cachexia | 7 | 77 | 1458 |
| chds | 4 | 500 | 24 |
| coronary | 6 | 1841 | 64 |
| fall | 4 | 50000 | 64 |
| ksl | 9 | 1083 | 512 |
| mathmarks | 5 | 88 | 243 |
| phd | 6 | 915 | 144 |
| titanic | 4 | 2201 | 32 |

**Table 1** Details about the datasets used in the computational study of Section 5.2.

| | BN | Marg. | Total | BHC | Simpl. | Greedy Marg. | All Marg. | All Total |
|---|---|---|---|---|---|---|---|---|
| asia | *22214.59* | 26225.00 | 22255.09 | **22174.53** | 22350.63 | 22500.89 | | |
| cachexia | 963.98 | 984.78 | 1121.77 | **791.98** | 1092.72 | *936.48* | | |
| *chds* | 2831.56 | 2826.09 | 2825.67 | **2824.87** | 2834.70 | 2834.32 | *2825.61* | *2825.61* |
| *coronary* | 13442.02 | 13972.00 | 13362.36 | **13292.00** | 13369.28 | 13457.04 | 13332.06 | *13322.98* |
| *fall* | 137807.93 | 138182.64 | 137495.12 | **137442.58** | 137536.66 | 139498.47 | 137468.30 | *137461.77* |
| ksl | *11801.14* | 12194.85 | 12358.51 | **11414.35** | 12319.85 | 12060.13 | | |
| *mathmarks* | 956.03 | 939.16 | 958.50 | **859.05** | 1024.04 | 934.73 | *923.97* | 958.50 |
| phd | 8372.93 | 8787.46 | 8528.46 | **8297.50** | 8526.54 | 8392.60 | *8360.52* | 8405.01 |
| *titanic* | 10502.28 | 10605.18 | 10450.01 | **10432.84** | 10453.93 | 10502.50 | 10453.93 | *10443.52* |

**Table 2** BIC for models learned over nine datasets: Bayesian networks (BN), simple staged trees with marginal algorithm (Marg.), simple staged trees with total algorithm (Total), generic staged trees with backward hill-climbing (BHC), simple staged trees derived via simplification (Simpl.), simple staged tree with variable ordering (Greedy Marg.), simple staged tree considering all orders and marginal algorithm (All-Marg.), simple staged tree considering all orders and total algorithm (All-Tot.).

| | BN | Marg. | Total | BHC | Simpl. | Greedy Marg. | All Marg. | All Total |
|---|---|---|---|---|---|---|---|---|
| asia | 32 | **12** | 18 | 38 (0.37) | 38 | 16 | | |
| cachexia | 25 | 17 | **10** | 66 (0.30) | 66 | 17 | | |
| chds | 8 | 7 | 7 | 8 (0.88) | 8 | **6** | 7 | 7 |
| coronary | 21 | 13 | 15 | 27 (0.59) | 27 | **11** | 14 | 12 |
| fall | 17 | 11 | 14 | 19 (0.68) | 19 | 10 | **9** | 10 |
| ksl | 37 | 16 | **14** | 168 (0.19) | 168 | 17 | | |
| mathmarks | 13 | 12 | **7** | 38 (0.39) | 38 | 11 | 10 | **7** |
| phd | 25 | **9** | 10 | 45 (0.33) | 45 | 12 | 13 | 10 |
| titanic | 21 | **10** | 16 | 17 (0.76) | 17 | **10** | 11 | 15 |

**Table 3** Number of positions for the staged trees reported in Table 2. For BHC, the number in parentheses is the ratio between the number of stages and the number of positions.

model is learned using the hill-climbing algorithm implemented in the `bnlearn` R package (Scutari, 2010) (labeled BN). A topological order of the learned BN is then used for learning simple staged trees using both the marginal and the total algorithm (labeled Marg. and Total, respectively). Such an order is also used to learn a more general staged tree using the backward hill-climbing algorithm in the `stagedtrees` R package (labeled BHC). The resulting staged tree is further simplified to give a simple staged tree (labeled Simpl.). Staged trees without assuming a fixed variable-ordering are also learned using the greedy-marginal algorithm discussed in Section 4.1 (labeled WO-Marg.). Lastly, if computationally feasible, we implemented an exhaustive model search using the marginal and total algorithm for every possible variable ordering (labeled All-Marg. and All-Tot., respectively).

| | BN | Marg. | Total | BHC | Simpl. | Greedy Marg. | All Marg. | All Total |
|---|---|---|---|---|---|---|---|---|
| asia | 0.014 | 0.084 | 0.534 | 0.955 | 0.021 | 0.133 | | |
| cachexia | 0.005 | 0.217 | 0.247 | 3.541 | 0.080 | 0.280 | | |
| chds | 0.003 | 0.015 | 0.024 | 0.050 | 0.002 | 0.019 | 0.219 | 0.487 |
| coronary | 0.007 | 0.034 | 0.168 | 0.770 | 0.003 | 0.061 | 16.097 | 69.568 |
| fall | 0.022 | 0.033 | 0.473 | 0.105 | 0.003 | 0.074 | 0.896 | 6.022 |
| ksl | 0.010 | 0.115 | 0.149 | 134.996 | 0.052 | 0.156 | | |
| mathmarks | 0.002 | 0.060 | 0.049 | 1.761 | 0.006 | 0.093 | 4.347 | 4.400 |
| phd | 0.004 | 0.031 | 0.106 | 3.934 | 0.008 | 0.060 | 21.241 | 39.688 |
| titanic | 0.005 | 0.021 | 0.125 | 0.097 | 0.002 | 0.024 | 0.628 | 1.682 |

**Table 4** Time (in seconds) to learn the models reported in Table 2. Computations carried out with an intel CORE i7 vPro 8th Gen.

The BIC of the learned models is reported in Table 2. The staged tree learned with the BHC algorithm has the lowest BIC for all datasets. This is expected since it searches over the largest model space, which includes those of all other models considered. We can further notice that for seven out of the nine datasets, one simple staged tree outperforms the BN model, as it can be seen from the second-highest scoring model reported in italics in Table 2. In five cases (names in italics) a staged tree learned with a fixed order (Marg., Total, or Simpl.) outperforms the BN model. This highlights that although simple staged trees have a highly constrained topology, they can still provide a good representation of real-world data by embedding non-symmetric conditional independences. Furthermore, it can be noticed that the BIC of models learned with a fixed-variable ordering is not too different from those of models learned without assuming an ordering a priori.

Although the BHC algorithm returns the best fitting model, it also learns staged trees whose equivalent CEG has a large number of vertices, as reported in Table 3. For all datasets, simple staged trees learned directly from data have a smaller number of positions, and therefore, the associated CEG representation is much more compact. Table 3 further reports the ratio between the number of stages and the number of positions for the staged tree learned with the BHC algorithm. The lower the ratio, the less effective the simplification process is (by construction, the ratio for the other trees is one). This highlights the need for algorithms that learn the structure of a simple staged tree from data since otherwise, the coalescence of the tree into the CEG merges a very small number of vertices: the resulting CEG would not be any simpler to graphically investigate than the original staged tree.

Table 4 reports the computational times for all models learned. While the BN is the model that requires less time, simple staged trees learned using the Marg. and Total algorithms require fractions of seconds. Furthermore, they are much faster than the BHC algorithm for generic staged trees coupled with the simplification process. Lastly, it can be noticed that the Greedy Marg. algorithm without a fixed order is fast and comparable to the Marg. and Total algorithms, despite having to consider multiple variable orderings.

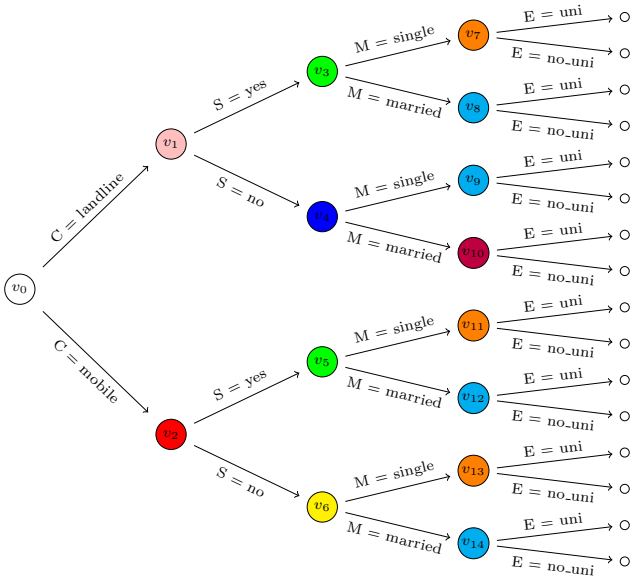



**Fig. 7** Simple staged tree learned with the All Marg. algorithm over the bank marketing dataset.

# 6 Applications

## 6.1 The Bank Marketing Dataset

Our first application studies a subset of the `bank` datset from the UCI machine learning repository reporting information on direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls with the purpose of making customers subscribe to a bank term deposit.

For illustrative purposes, we consider only four binary variables: marital status (M = single/married); education (E = uni/no_uni); contact (C = mobile/landline); subscription (S = yes/no). For each customer, we consider only the first time they were contacted by the bank.

The best scoring BN embeds one conditional independence: $\mathrm{S} \perp\!\!\!\perp \mathrm{M}|\mathrm{E}, \mathrm{C}$. Whether a customer subscribes to the bank product is independent of her marital status given her educational level and the type of contact. Simple staged trees are learned with all algorithms considered so far, and it can be shown that they all outperform the BN in terms of model fit. Here, we consider the simple staged tree learned with the All Marg. algorithm. This is reported in Figure 7 and it has variable ordering $(\mathrm{C}, \mathrm{S}, \mathrm{M}, \mathrm{E})$. From the staging $\{v_3, v_5\}$ it follows that $\mathrm{M} \perp\!\!\!\perp \mathrm{C}|\mathrm{S} = \text{yes}$. The staging over the vertices $v_{11}, \ldots, v_{14}$ implies that $\mathrm{E} \perp\!\!\!\perp \mathrm{S}|\mathrm{M}, \mathrm{C} = \text{mobile}$. The staging over the vertices $v_7, \ldots, v_9$ implies additional highly non-symmetric independences which in general cannot be depicted by BNs.

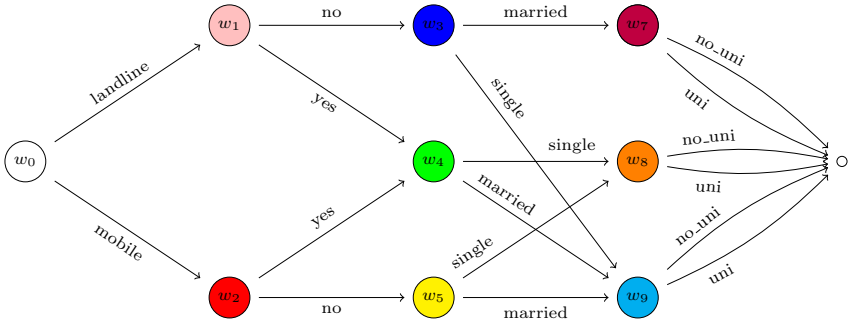



**Fig. 8** The CEG equivalent to the staged tree in Figure 7 for the bank marketing dataset.

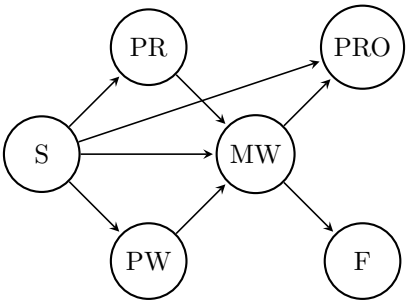



**Fig. 9** BN learned with hill-climbing for the coronary dataset.

The equivalent CEG is reported in Figure 8. Since the staged tree is simple each stage exactly corresponds to a vertex in the CEG, which are colored as in the tree in Figure 7 for ease of understanding. There are two paths that lead to the position $w_4$ coinciding with a subscriber to the bank product for both landline and mobile contact. This represents the already noticed context-specific independence $\mathrm{M} \perp\!\!\!\perp \mathrm{C}|\mathrm{S} = \text{yes}$. Furthermore, it also implies that the conditional distribution of subsequent random variables, in this case E, is the same for $(\mathrm{C} = \text{mobile}, \mathrm{S} = \text{yes})$ and $(\mathrm{C} = \text{landline}, \mathrm{S} = \text{yes})$. Let's consider position $w_9$: there are four paths leading to this vertex which represent contexts for the variables C, S, and M - namely (landline, no, single), (landline, yes, married), (mobile, yes, married) and (mobile, no, married). The conditional distribution of education is the same for these four contexts.

Although, for this application, the sample space is small enough that we can read conditional independence directly from the staged tree, it is apparent that the CEG already provides a more compact representation of the model. The compactness of the learned CEG is maximized since the equivalent staged tree has been learned from data to be simple.

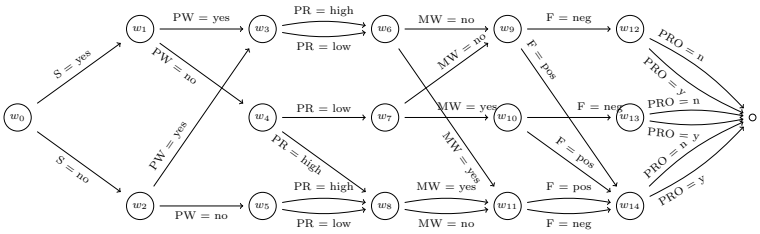



**Fig. 10** The CEG equivalent to the simple staged tree learned over the coronary dataset with the total algorithm.

## 6.2 The Coronary Dataset

Next, we consider the coronary dataset included in `bnlearn` reporting risk factors of coronary thrombosis of 1841 men. The dataset includes six binary variables: smoking (S=yes/no); strenuous mental work (MW=yes/no); strenuous physical work (PW=yes/no); systolic blood pressure (PR=high/low); family anamnesis of coronary heart disease (F=pos/neg); ratio of beta and alpha lipoproteins (PRO=y/n). The BN model learned with hill-climbing is reported in Figure 9 and embeds three conditional independence statements: $\text{PR} \perp\!\!\!\perp \text{PW}|\text{S}$, $\text{F} \perp\!\!\!\perp \text{S}, \text{PR}, \text{PW}|\text{MW}$ and $\text{PRO} \perp\!\!\!\perp \text{F}, \text{PR}, \text{PW}|\text{MW}, \text{S}$.

The simple staged tree learned with the Total algorithm provides a better fit to the data (see Table 2). However, it becomes challenging to visualize since it has 63 internal nodes and 64 leaves. Conversely, the equivalent CEG has 15 internal nodes and is reported in Figure 10. Since the equivalent staged tree is simple, each stage of the tree coincides with a vertex of the CEG. Notice that the CEG equivalent to the staged tree learned with backward hill-climbing would have more than 30 vertices (see Table 3) and would therefore be very challenging to interpret.

The CEG in Figure 10 embeds a wide array of non-symmetric independences that BNs cannot graphically report. For instance the edges $(w_1, w_3)$ and $(w_2, w_3)$ are associated to the context-specific independence $\text{PR} \perp\!\!\!\perp \text{S}|\text{PW} = \text{yes}$. The edges between $w_3$ and $w_6$ again represent the context-specific independence $\text{MW} \perp\!\!\!\perp \text{S}, \text{PR}|\text{PW} = \text{yes}$ and the edges between $w_5$ and $w_8$ are associated to $\text{MW} \perp\!\!\!\perp \text{PR}|\text{S} = \text{no}, \text{PW} = \text{no}$. The edge $(w_4, w_8)$ can be described as what is usually termed a local independence: the conditional distribution of MW given S = no and PW = no is equal to the conditional distribution of MW given S = yes, PW = no and PR = high. Lastly, we can notice that many edges lead to the position $w_{14}$. This tells us that the conditional distribution of PRO is equal given any conditioning of the preceding variables which can be read from the labels of edges in paths that end in $w_{14}$. This further implies that given F = pos, PRO is independent of all other variables.

# 7 Conclusions

Staged trees and CEGs provide a flexible framework to graphically represent any non-symmetric independence existing in a random vector. However, for

generic data-learned staged trees, the conversion of the staged tree into the equivalent CEG representation does not provide a significant simplification of the underlying graph. In this paper, we introduced novel structural learning algorithms for the class of simple staged trees for which the conversion into a CEG drastically reduces the number of vertices. Although the requirement of learning a simple staged tree greatly reduces the size of the model search, our experiment demonstrated that simple staged trees often outperform BNs in model fit. The insights that the CEG provides into the dependence structure have been highlighted using two datasets.

We thus claim that if the objective is to learn a CEG, the appropriate search space to explore is the one of the simple staged trees. Otherwise, the CEG graphical representation can, in general, be avoided since it will not represent a more compact graph with respect to the staged tree.

The algorithms proposed here only search the space of simple staged trees. One possible extension is to consider searching algorithms over the whole model space of staged trees which strongly penalize staging structures leading to CEGs with many vertices. The learned staged tree would not necessarily be simple but still, be such that the equivalent CEG provides a significant compact representation of the model. The development of such algorithms is the focus of current research.

The two data applications illustrated how conditional independences can be read from a learned CEG model. In practice, this entails investigating all paths leading to a specific position and their edge labels. Thwaites and Smith (2015) introduced a criterion to read non-symmetric conditional independences directly from the CEG, but its application can be at first challenging. We are planning to implement this criterion in the next release of the `stagedtrees` R package, mimicking the usage of the `dsep` `bnlearn` function to check symmetric conditional independences in BNs (Scutari, 2010).

## Declarations

All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

Gherardo Varando's work was funded by the European Research Council (ERC) Synergy Grant "Understanding and Modelling the Earth System with Machine Learning (USMILE)" under Grant Agreement No 855187.

## References

Barclay, L.M., Hutton, J.L., Smith, J.Q. (2013). Refining a Bayesian network using a chain event graph. *International Journal of Approximate Reasoning*, *54*, 1300-1309.

Boutilier, C., Friedman, N., Goldszmidt, M., Koller, D. (1996). Context-specific independence in Bayesian networks. *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence* (p. 115-123).

Cano, A., Gómez-Olmedo, M., Moral, S., Pérez-Ariza, C.B., Salmerón, A. (2012). Learning recursive probability trees from probabilistic potentials. *International Journal of Approximate Reasoning*, *53*(9), 1367-1387.

Carli, F., Leonelli, M., Riccomagno, E., Varando, G. (2022). The R package stagedtrees for structural learning of stratified staged trees. *Journal of Statistical Software*, *102*, 1–30.

Carli, F., Leonelli, M., Varando, G. (2023). A new class of generative classifiers based on staged tree models. *Knowledge-Based Systems*, *268*, 110488.

Chickering, D.M., Heckerman, D., Meek, C. (1997). A Bayesian approach to learning Bayesian networks with local structure. *Proceedings of 13th Conference on Uncertainty in Artificial Intelligence* (pp. 80–89).

Collazo, R.A., Görgen, C., Smith, J.Q. (2018). *Chain event graphs*. Chapmann & Hall.

Collazo, R.A., & Smith, J.Q. (2016). A new family of non-local priors for chain event graph model selection. *Bayesian Analysis*, *11*(4), 1165-1201.

Corander, J., Hyttinen, A., Kontinen, J., Pensar, J., Väänänen, J. (2019). A logical approach to context-specific independence. *Annals of Pure and Applied Logic*, *170*(9), 975–992.

Cowell, R.G., & Smith, J.Q. (2014). Causal discovery through MAP selection of stratified chain event graphs. *Electronic Journal of Statistics*, *8*(1), 965–997.

Daly, R., Shen, Q., Aitken, S. (2011). Learning Bayesian networks: Approaches and issues. *The Knowledge Engineering Review*, *26*(2), 99.

Darwiche, A. (2009). *Modeling and reasoning with Bayesian networks*. Cambridge University Press.

DesJardins, M., Rathod, P., Getoor, L. (2008). Learning structured Bayesian networks: Combining abstraction hierarchies and tree-structured conditional probability tables. *Computational Intelligence*, *24*(1), 1–22.

Duarte, E., & Solus, L. (2021). Representation of context-specific causal models with observational and interventional data. *arXiv:2101.09271*.

Duarte, E., & Solus, L. (2023). A new characterization of discrete decomposable graphical models. *Proceedings of the American Mathematical Society*, *151*(03), 1325–1338.

Fergusson, D., & Horwood, J. (2001). The Christchurch health and development study: Review of findings on child and adolescent mental health. *Australian & New Zealand Journal of Psychiatry*, *35*(3), 287–296.

Freeman, G., & Smith, J.Q. (2011). Bayesian MAP model selection of chain event graphs. *Journal of Multivariate Analysis*, *102*(7), 1152-1165.

Friedman, N., & Goldszmidt, M. (1996). Learning Bayesian networks with local structure. *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence* (pp. 252–262).

Geiger, D., & Heckerman, D. (1996). Knowledge representation and inference in similarity networks and Bayesian multinets. *Artificial Intelligence*, *82*, 45-74.

Geiger, D., Heckerman, D., King, H., Meek, C. (2001). Stratified exponential families: Graphical models and model selection. *Annals of Statistics*, *29*(2), 505–529.

Genewein, T., McGrath, T., Déletang, G., Mikulik, V., Martic, M., Legg, S., Ortega, P.A. (2020). Algorithms for causal reasoning in probability trees. *arXiv:2010.12237*.

Görgen, C., Bigatti, A., Riccomagno, E., Smith, J.Q. (2018). Discovery of statistical equivalence classes using computer algebra. *International Journal of Approximate Reasoning*, *95*, 167–184.

Görgen, C., Leonelli, M., Marigliano, O. (2022). The curved exponential family of a staged tree. *Electronic Journal of Statistics*, *16*(1), 2607–2620.

Görgen, C., Leonelli, M., Smith, J.Q. (2015). A differential approach for staged trees. *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty* (pp. 346–355).

Görgen, C., & Smith, J.Q. (2018). Equivalence classes of staged trees. *Bernoulli*, *24*(4A), 2676–2692.

Hyttinen, A., Pensar, J., Kontinen, J., Corander, J. (2018). Structure learning for Bayesian networks over labeled DAGs. *International Conference on Probabilistic Graphical Models* (pp. 133–144).

Jaeger, M., Nielsen, J.D., Silander, T. (2006). Learning probabilistic decision graphs. *International Journal of Approximate Reasoning*, *42*(1-2), 84-100.

Leonelli, M. (2019). Sensitivity analysis beyond linearity. *International Journal of Approximate Reasoning*, *113*, 106–118.

Leonelli, M., & Varando, G. (2022). Highly efficient structural learning of sparse staged trees. *International Conference on Probabilistic Graphical Models* (pp. 193–204).

Leonelli, M., & Varando, G. (2023a). Context-specific causal discovery for categorical data using staged trees. *International Conference on Artificial Intelligence and Statistics* (pp. 8871–8888).

Leonelli, M., & Varando, G. (2023b). Learning and interpreting asymmetry-labeled DAGs: A case study on COVID-19 fear. *arXiv:2301.00629*.

Leonelli, M., & Varando, G. (2024). Robust learning of staged tree models: A case study in evaluating transport services. *arXiv:2401.01812*.

Neapolitan, R.E. (2004). *Learning Bayesian networks.* Pearson Prentice Hall.

Nicolussi, F., & Cazzaro, M. (2021). Context-specific independencies in stratified chain regression graphical models. *Bernoulli*, *27*(3), 2091–2116.

Pearl, J. (2009). *Causality.* Cambridge University Press.

Pensar, J., Nyman, H., Koski, T., Corander, J. (2015). Labeled directed acyclic graphs: A generalization of context-specific independence in directed graphical models. *Data Mining and Knowledge Discovery*, *29*(2), 503–533.

Pensar, J., Nyman, H., Lintusaari, J., Corander, J. (2016). The role of local partial independence in learning of Bayesian networks. *International Journal of Approximate Reasoning*, *69*, 91–105.

Poole, D., & Zhang, N. (2003). Exploiting contextual independence in probabilistic inference. *Journal of Artificial Intelligence Research*, *18*, 263-313.

Salmerón, A., Cano, A., Moral, S. (2000). Importance sampling in Bayesian networks using probability trees. *Computational Statistics and Data Analysis*, *34*(4), 387-413.

Scutari, M. (2010). Learning Bayesian networks with the bnlearn R package. *Journal of Statistical Software*, *35*(3), 1–22.

Shafer, G. (1996). *The art of causal conjecture.* MIT Press.

Shen, Y., Choi, A., Darwiche, A. (2020). A new perspective on learning context-specific independence. *Proceedings of the 10th International Conference on Probabilistic Graphical Models* (pp. 425–436).

Shenvi, A., & Smith, J.Q. (2020). Constructing a chain event graph from a staged tree. *Proceedings of the 10th International Conference on Probabilistic Graphical Models* (pp. 437–448).

Silander, T., & Leong, T. (2013). A dynamic programming algorithm for learning chain event graphs. *Proceedings of the International Conference on Discovery Science* (p. 201-216).

Smith, J.Q., & Anderson, P.E. (2008). Conditional independence and chain event graphs. *Artificial Intelligence*, *172*(1), 42 - 68.

Thwaites, P.A., & Smith, J.Q. (2015). A separation theorem for chain event graphs. *arXiv:1501.05215*.

Thwaites, P.A., Smith, J.Q., Cowell, R.G. (2008). Propagation using chain event graphs. *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence* (pp. 546–553).

Tikka, S., Hyttinen, A., Karvanen, J. (2019). Identifying causal effects via context-specific independence relations. *Advances in Neural Information Processing Systems* (pp. 2804–2814).

Tsagris, M. (2020). A new scalable Bayesian network learning algorithm with applications to economics. *Computational Economics*, 1–27.

Varando, G., Carli, F., Leonelli, M. (2021). Staged trees and asymmetry-labeled DAGs. *arXiv:2108.01994*.

Wang, Z., Gao, X., Yang, Y., Tan, X., Chen, D. (2020). Learning Bayesian networks based on order graph with ancestral constraints. *Knowledge-Based Systems*, *211*, 106515.

Yang, Y., Gao, X., Guo, Z., Chen, D. (2019). Learning Bayesian networks using the constrained maximum a posteriori probability method. *Pattern Recognition*, *91*, 123–134.