Regular vines with strongly chordal pattern of (conditional) independence

Zhu, Kailun; Kurowicka, Dorota

DOI
10.1016/j.csda.2022.107461

Publication date
2022

Document Version
Final published version

Published in
Computational Statistics and Data Analysis

Computational Statistics and Data Analysis 172 (2022) 107461

Contents lists available at ScienceDirect

Computational Statistics and Data Analysis

www.elsevier.com/locate/csda

Regular vines with strongly chordal pattern of (conditional) independence

Kailun Zhu, Dorota Kurowicka *

DIAM, Delft University of Technology, Delft, the Netherlands

ARTICLE INFO

Article history:
Received 2 August 2021
Received in revised form 2 March 2022
Accepted 2 March 2022
Available online 15 March 2022

Keywords:
Conditional independence
Strongly chordal graph
Regular vine copula

ABSTRACT

Multivariate statistical models can be simplified by assuming that a pattern of conditional independence is presented in the given data. A popular way of capturing the (conditional) independence is to use probabilistic graphical models. The relationship between strongly chordal graphs and m-saturated vines is proved. Moreover, an algorithm to construct an m-saturated vine structure corresponding to strongly chordal graph is provided. This allows the reduction of regular vine copula models complexity. When the underlying data is sparse our approach leads to model estimation improvement when compared with current heuristic methods. Furthermore, due to reduction of model complexity it is possible to evaluate all vine structures as well as to fit non-simplified vines. These advantages have been shown in the simulated and real data examples.[1]

1. Introduction

Probabilistic graphical models are very useful in representing the joint distribution of random variables. In these models the variables are associated with the vertices of the graph and the (conditional) independence in the joint distribution of these variables are determined by the absence of edges. It suffices to study the structure of the graph in order to assess whether variables are (conditionally) independent. The corresponding joint density can then be built with additional information provided on the types of dependence encoded in the edges. For example, the joint probability density represented by a chordal graph is known to be equal to the product of densities of the variables in maximal cliques divided by the product of densities over the variables in the separators (Pearl, 1988), and the joint density of a Bayesian network is the product of conditional densities of variables given their parents in the graph (Lauritzen, 1996). However, for continuous random variables, these models are not easy to be estimated or/and are not simple to be sampled in high dimensions unless we assume that the distributions are multivariate Gaussian or the dependence is encoded by Gaussian copulas.

The regular vine model, which is a set of nested trees, is another example of graphical model to represent a joint distribution. It allows to decompose a n-dimensional density into n marginal densities, $n-1$ copulas and $\binom{n-1}{2}$ conditional copulas. All bivariate (conditional) copulas are algebraically independent and can be chosen from any parametric family or can be assumed to be non-parametric. As proved in Nápoles (2010) there are $\frac{n!}{2}2^{\binom{n-2}{2}}$ regular vine structures for n variables. Each vine structure represents one way of decomposing the joint density (Bedford and Cooke, 2002; Kurowicka and Joe,

* Corresponding author.
E-mail addresses: K.Zhu-3@tudelft.nl (K. Zhu), D.Kurowicka@tudelft.nl (D. Kurowicka).

[1] All R code files involving algorithms in this paper are attached as online supplementary material.

https://doi.org/10.1016/j.csda.2022.107461

2011; Czado, 2019). A popular heuristic, proposed in Dißmann et al. (2013), suggests to find a vine structure for the data tree by tree. Each tree structure is determined by a maximum spanning tree with weights being the absolute Kendall's tau. The estimation of parameters in this model is performed tree-wise, which is less expensive than optimizing simultaneously over all parameters (Aas et al., 2009). Furthermore, in applications, regular vines are used in their simplified form (simplified vines), where conditional copulas do not depend directly on the conditioning variables. A detailed discussion about the simplifying assumption can be found in Stöber et al. (2013), Spanhel and Kurz (2015), Kurz and Spanhel (2017). Moreover, the distribution represented by a simplified regular vine model can be sampled very efficiently. These procedures are implemented in the **VineCopula** package (Nagler et al., 2019) in R.

In principle regular vines are fully connected graphs and the (conditional) independence can be specified by setting the bivariate copulas to be the independence copula. This can be seen as removing some nodes in the nested set of trees. In general, however, it is not known how the structure of a regular vine with some nodes missing can be used to describe all (conditional) independence of the random variables in the joint distribution. There are cases where the conditional independence can be easily specified in a regular vine model. In Brechmann et al. (2012), truncated regular vines are introduced where all bivariate copulas above certain tree level are set to be the independence copula. The level of truncation is chosen during the tree-wise estimation process (estimation stops when the absolute AIC of the new copulas in the tree is smaller than a predetermined threshold or the improvement in likelihood by adding the next tree is not sufficient (determined by the Vuong test (Vuong, 1989)). Determining (conditional) independence during estimation is convenient but is in general not optimal. The chosen vine structure, the simplifying assumption and the tree-wise estimation procedure can result in wrongly specified conditional (in)dependence in the estimated model.

The idea of simplifying regular vine copula models by assuming a pattern of conditional independence has been already discussed in the literature. Both Müller and Czado (2018) and Hobæk Haff et al. (2016) present conditions under which a directed acyclic graph (DAG) in the former and a chordal graph in the latter corresponds to a truncated vine. Then heuristics are provided to construct the vine copula model based on a graph obtained from data. A different heuristic construction of the vine structure is proposed in Müller and Czado (2019b), where conditional independence result from structural equation models. In Müller and Czado (2019a) the conditional independence is found using graphical Lasso (Friedman et al., 2007), where variables are partitioned into homogeneous groups and subvines corresponding to each group are estimated at first. Then these subvines are combined together in a similar way as by using the concept of merging discussed in Cooke et al. (2015). This method requires to specify a pre-determined level for a maximum number of dimensions for subvines as well as a truncation level.

In this paper we show that the set of conditional independence in the form of a special chordal graph, called strongly chordal graph (Farber, 1983), can be represented by regular vine copulas. Our main result, in this paper, establishes the equivalence between strongly chordal graphs and m-saturated vines (abbreviated as m-vine, introduced in Kurowicka and Cooke, 2006). A special case of this relationship has been studied in Hobæk Haff et al. (2016), where the authors showed that the truncated vines (which are special cases of m-vines) correspond to chordal graphs satisfying extra conditions.

The paper is organized as follows: we start with basic notations concerning graph theory and regular vines in Section 2; In Section 3 the main theorem of this paper describing the relationship between a strongly chordal graph and an m-vine is proved and an algorithm to construct an m-vine corresponding to a given graph is proposed. Simulation results are presented in Section 4, where two examples showing the advantages of the m-vine approach are listed in Section 4.1. A general heuristic of choosing the m-vine structure for a given data set is proposed in Section 4.2, which is tested by a simulation study in Section 4.3. Finally, a real data analysis is implemented in Section 5 and the conclusions finalize the paper in Section 6.

2. Background

In this section, we present basic definitions from graph theory and a brief introduction to regular vines and regular vine copulas, as well as the notation we will use in the paper.

2.1. Graphs

Let $\mathcal{G}$ be a graph with vertices $\mathcal{V}(\mathcal{G}) = \{v_1, \ldots, v_n\}$ and edges $\mathcal{E}(\mathcal{G})$. A graph is **complete** if every pair of vertices is connected by a unique edge. Two vertices are **adjacent** if there is an edge between them, whereas the adjacency of edges means two edges share a common vertex. We say that $\mathcal{G}(\mathcal{U})$ is a **subgraph** of graph $\mathcal{G}$ induced by the vertex set $\mathcal{U}$ if $\mathcal{G}(\mathcal{U})$ is the graph with vertex set $\mathcal{U} \subseteq \mathcal{V}(\mathcal{G})$ and with edge set $\{(u, v) \in \mathcal{E}(\mathcal{G}) | u, v \in \mathcal{U}\}$. $\mathcal{C} \subseteq \mathcal{V}(\mathcal{G})$ is a **clique** of $\mathcal{G}$ when $\mathcal{G}(\mathcal{C})$ is a complete subgraph. If $\mathcal{G}(\mathcal{C})$ is a maximal complete subgraph of graph $\mathcal{G}$ then $\mathcal{C}$ is called **maximal clique**.

A **path** of length k between vertices α and β in graph $\mathcal{G}$ is a sequence $\alpha = \alpha_0, \ldots, \alpha_k = \beta$ of distinct vertices of $\mathcal{G}$ such that $(\alpha_{i-1}, \alpha_i) \in \mathcal{E}(\mathcal{G})$ for all $i = 1, \ldots, k$. A **cycle** of length k is a path of length k in which the end points are identical ($\alpha = \beta$).

A vertex set $\mathcal{S} \subseteq \mathcal{V}(\mathcal{G})$ is said to be (α, β)**-separator** if all paths from α to β intersect $\mathcal{S}$. Furthermore the set $\mathcal{S}$ is said to **separate** vertex set $\mathcal{A}$ from vertex set $\mathcal{B}$ if $\mathcal{S}$ is an (α, β)-separator for every $\alpha \in \mathcal{A}$ and $\beta \in \mathcal{B}$.

A connected graph T is called a **tree** if it has no cycle.

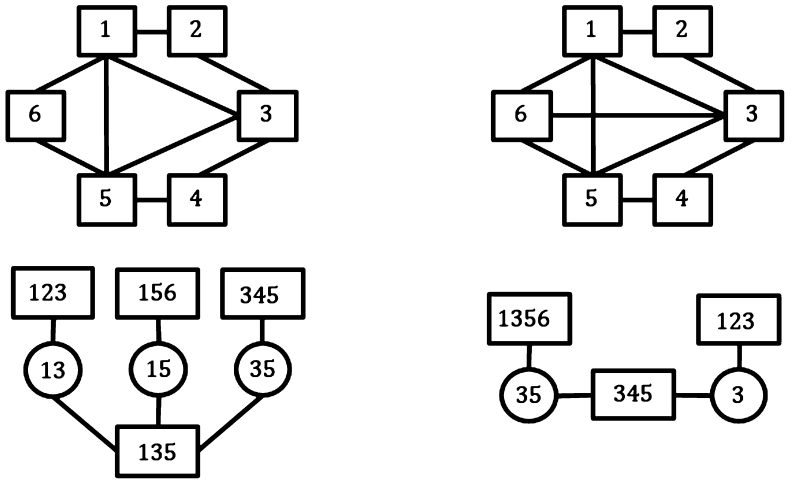

Fig. 1. The graphs below are the clique trees of chordal graphs above. The left graph is a chordal graph but not strongly chordal while the right one is a strongly chordal graph.

Graph $\mathcal{G}$ is said to be **chordal** if every cycle of $\mathcal{G}$ with length larger than 3 has a chord (where chord of a cycle is an edge joining two nonconsecutive vertices in the cycle). Hence, a tree is an example of a chordal graph. A **strongly chordal** graph is a chordal graph with the property that every cycle of even length of at least six contains a chord. This chord combines with the edges of the cycle to form two shorter even-length cycles.

In this paper we will discuss also graphs whose vertices correspond to sets of elements (and vertices will be referred to as nodes) and edges correspond to intersections of sets. A clique tree (or junction tree), $\mathcal{T}$, of graph $\mathcal{G}$ is an example of such a graph. If $\boldsymbol{\mathcal{C}}(\mathcal{G}) = (\mathcal{C}_1, ..., \mathcal{C}_k)$ is a set of maximal cliques of graph $\mathcal{G}$ then $\mathcal{T}$ with nodes $\boldsymbol{\mathcal{C}}(\mathcal{G})$ is said to be a **clique tree** if any nonempty intersection $\mathcal{C}_i \cap \mathcal{C}_j$ is contained in every node on the unique path in $\mathcal{T}$ between $\mathcal{C}_i$ and $\mathcal{C}_j$. This property is called the running intersection property (RIP). The set of separators $\boldsymbol{\mathcal{S}}(\mathcal{T})$ in a clique tree is the nonempty intersections of adjacent nodes. For a vertex v of $\mathcal{G}$, $\mathcal{T}_{\{v\}}$ denotes the subgraph of $\mathcal{T}$ induced by nodes containing v. If $\mathcal{T}$ is a clique tree, then for any $v \in \mathcal{V}(\mathcal{G})$, $\mathcal{T}_{\{v\}}$ is always connected according to RIP.

We show below an example (from Mukhopadhyay and Rahman (2021)) to illustrate the above definitions.

In Fig. 1, two graphs with six nodes (above) and their corresponding clique trees (underneath) are presented. Maximal cliques are shown in squares and separators are shown in circles in the clique tree. The left graph contains four cliques $\{1, 2, 3\}$, $\{1, 5, 6\}$, $\{3, 4, 5\}$ and $\{1, 3, 5\}$ (this can be seen also in its clique tree). This graph is chordal but it is not strongly chordal. This graph can be made strongly chordal when an extra edge 36 is added. Alternatively, edge 14 or edge 25 instead of edge 36 could be added. The graph constructed by adding edge 36 is shown in Fig. 1 (right). We can observe that the addition of edge 36 leads to the existence of clique $\{1, 3, 5, 6\}$, which is composed of two cliques $\{1, 5, 6\}$ and $\{1, 3, 5\}$.

In the next section, a graphical model called regular vine introduced in Bedford and Cooke (2001, 2002) is presented.

2.2. Vines

A vine on n elements (referred to as $V(n)$) or on element set $\mathcal{V}_n = \{v_1, \ldots, v_n\}$ ($V(\mathcal{V}_n)$), is a nested set of connected trees $V = \{T_1, ..., T_{n-1}\}$ where the edges of tree T_j are the nodes of tree T_{j+1}, $j = 1, \ldots, n-2$. Vines are in principle fully connected graphs. Such a graphical structure is quite useful and flexible in representing the dependence structure of a given data as shown in Kurowicka and Joe (2011); Czado (2019). We first define regular vines, which we call complete vines, and then introduce incomplete vines, obtained when some nodes in the regular vine are removed.

2.2.1. Complete vines

A regular vine is a set of connected trees in which two edges in tree T_j are joined by an edge in tree T_{j+1} only if these edges share a common node, $j = 1, ..., n-2$. The formal definition is as follows.

Definition 2.1 *(Regular vine).* V is a regular vine if

1. $V = \{T_1, \ldots, T_{n-1}\}$,
2. T_1 is a connected tree with nodes $N_1 = \mathcal{V}(T_1)$, and edges $E_1 = \mathcal{E}(T_1)$;
   for $i = 2, \ldots, n-1$ T_i is a tree with nodes $N_i = E_{i-1}$.
3. (proximity) For $i = 2, \ldots, n-1$, $\{a, b\} \in E_i$, $\#a \triangle b = 2$ where $\triangle$ denotes the symmetric difference.

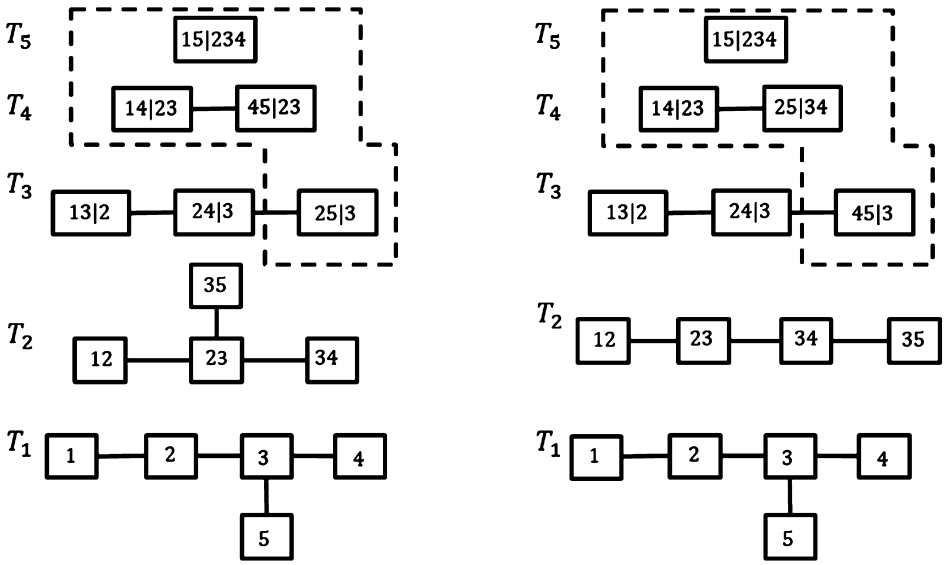

Fig. 2. Two regular vines on 5 elements with conditioned and conditioning sets. The left vine is denoted as $V_1(5)$ and the right one is $V_2(5)$. The nodes included in the dashed areas can be removed to form incomplete vines.

The proximity condition can be interpreted using graph theory terminology as follows: the node in tree T_i that connects two nodes in tree T_{i-1} is called a m-parent (parent in short) node and these two nodes are called the m-children (children in short) of the parent node, or siblings. Since all trees in a vine are connected, then each node in T_i has a sibling, and the proximity condition implies that each node has a common child with its sibling.

In the definition above we see that edges of tree T_i are nodes of tree T_{i+1}. To make the exposition more clear, in this paper we will always refer to nodes of a tree in a vine (which is consistent with the terminology introduced in Section 2.1 as the nodes in trees higher than T_1 can be associated with their constraint set that are subsets of vertices. Although the nodes in the first tree are just the vertices when we discuss vines we will refer to them as nodes). Hence, for the purpose of completeness, the single edge in tree T_{n-1} is referred to as a single node of tree T_n, and V is a set of n trees $V = \{T_1, \ldots, T_n\}$ where T_n is just one node called the **top node** of V.

Regular vines are used as graphical structures that allow the choice of a set of algebraically independent bivariate (conditional) copulas to factorize a joint density. This is done via bivariate conditional constraints associated with edges in each tree, which requires definition of **constraint**, **conditioned** and **conditioning sets** shown below.

Definition 2.2 *(Constraint, conditioned, and conditioning sets).*

1. For $e \in N_i, i \leq n$, the **constraint set** associated with a node e is the **complete union** U_e^* of e, that is, the subset of $\mathcal{V}(T_1)$ reachable from e by the membership relation.
2. For $i = 1, \ldots, n-1, e \in E_i$, if $e = \{j, k\}$ then the **conditioning set** associated with e is

$$D_e = U_j^* \cap U_k^*$$

and the **conditioned set** associated with e is

$$\{C_{e,j}, C_{e,k}\} = \{U_j^* \setminus D_e,\ U_k^* \setminus D_e\}.$$

We can see that for a node $e \in N_2$ the conditioning set is empty. Often a node e in a regular vine is associated with its conditioned and conditioning sets and we write $e = \{C_{e,j}, C_{e,k}|D_e\}$ or simply with its constraint set U_e^*. According to the proximity condition, the cardinality of the conditioned set is always 2 and $C_{e,j}, C_{e,k}$ are called **partners**. Fig. 2 shows two examples of a regular vine on 5 elements, where all 5 trees are shown and the conditioned and conditioning sets are printed in each node. These vines are denoted as $V_1(5)$ (left) and $V_2(5)$ (right).

The m-descendant/m-ancestor (descendant/ancestor in short) relationship in regular vine is defined as follows,

Definition 2.3 *(Descendant; ancestor).* If a node e is reachable from a node f via the membership relation: $e \in e_1 \in \cdots \in f$, we say that e is a descendant of f and f is an ancestor of e.

For $V_1(5)$ in Fig. 2 (left), node 12 is a descendant of node 14|23 and 14|23 is an ancestor of 12 (as node 13|2 is a child of 14|23 and parent of 12). However 14|23 is not an ancestor of neither node 35 nor node 5. We can now introduce the

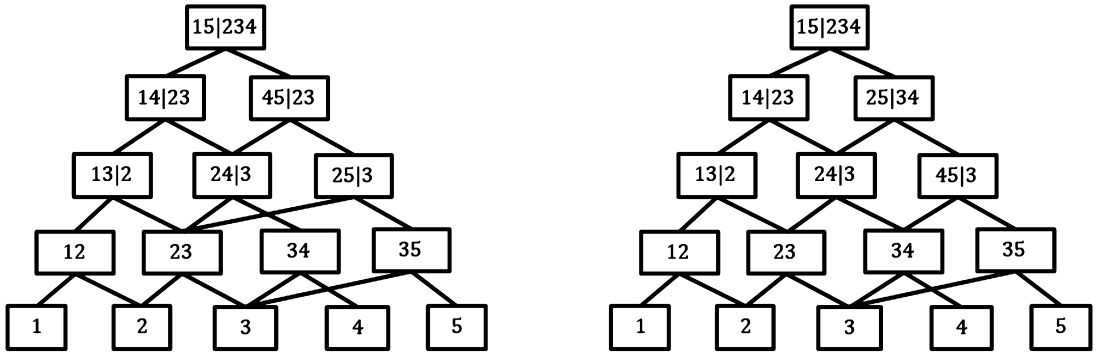

Fig. 3. Vine triangular array for $V_1(5)$ (left), and $V_2(5)$ (right) in Fig. 2.

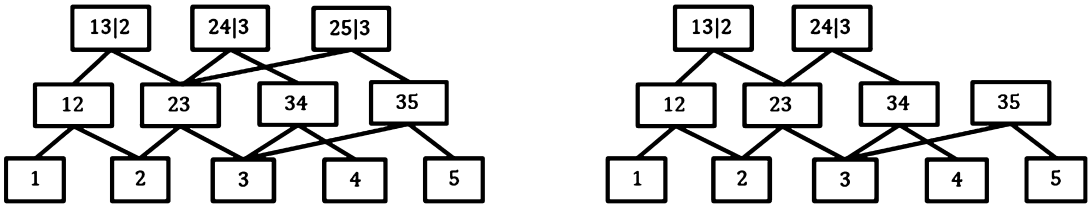

Fig. 4. Vine triangular arrays for the 3-truncated vine (left) of $V_1(5)$, and for the m-vine (right) obtained by removing the nodes in the dashed area in $V_2(5)$ in Fig. 2.

notion of a subvine of $V(\mathcal{V}_n)$. Given a node e with constraint set U_e^* in a regular vine $V(\mathcal{V}_n), U_e^* \subseteq \mathcal{V}_n$, the regular vine $V(U_e^*)$ is called a **subvine** of $V(\mathcal{V}_n)$ induced by the elements U_e^* if e is the top node of $V(U_e^*)$ and all descendants of e in $V(\mathcal{V}_n)$ belong to $V(U_e^*)$.

In order to make nodes and their parent-child relationships easily visible, we will use in this paper another representation of regular vines, called vine triangular array (Cooke et al., 2015). Nodes in each tree are shown in corresponding levels of the vine triangular array and the parent-child relationships are represented by a line connecting nodes. The vine triangular arrays of the vines $V_1(5)$ and $V_2(5)$ in Fig. 2 are shown in Fig. 3.

In the next section we will introduce incomplete vines.

2.2.2. Incomplete vines

Incomplete vines are vines from which some nodes have been removed. In this paper, two types of incomplete vines: truncated vines (Brechmann et al., 2012) and m-saturated vines (Kurowicka and Cooke, 2006), are considered.

Definition 2.4 *(Truncated vine).* An incomplete vine is a k-truncated vine of a regular vine $V(n)$, $1 \leq k \leq n$, if all nodes in trees $T_i, k+1 \leq i \leq n$ have been removed.

Definition 2.5 *(m-Saturated vine (short m-vine)).* An incomplete vine is a m-saturated vine of a regular vine $V(n)$ if all descendants of a node in its node set belong to the node set of the incomplete vine.

It is easy to observe that a truncated vine is a special type of m-vine.

Regular vines have only one top node, which is the node in the highest tree level. However, for truncated vines or m-vines there are several 'top' nodes, which we call **maximal nodes**. Maximal nodes are nodes that do not have ancestors. For example, the maximal nodes in the 3-truncated vine of $V_1(5)$, shown in Fig. 4 (left), are $13|2, 24|3, 25|3$. The maximal nodes of m-vine can appear in different tree levels. For $V_2(5)$, shown in Fig. 4 (right), the maximal nodes in the m-vine constructed by removing nodes in the dashed area are $13|2, 24|3, 35$.

We will represent an m-vine as $V(U_{e_1}^*, \ldots, U_{e_k}^*)$ where $U_{e_1}^*, \ldots, U_{e_k}^*$ are the constraint sets of the maximal nodes $e_1, \ldots, e_k$. Note that this representation is not unique as both m-vines in Fig. 2 obtained by removing nodes in dashed areas have the same maximal nodes, hence the same representation: $V(\{1,2,3\},\{2,3,4\},\{3,5\})$. We will call two maximal nodes **adjacent** if these two nodes share common nodes in their descendants (a common subvine). For example, we can observe that maximal nodes $24|3$, $13|2$ and 35 are adjacent.

In the next section we show how a joint distribution can be represented by assigning bivariate copulas to the nodes in $T_2, \ldots, T_n$ and marginal densities to the nodes of T_1.

2.3. Vine distribution

According to Sklar's theorem Sklar (1959) a joint density can be decomposed into the product of the marginal densities and a copula density. The copula can be further decomposed into a series of bivariate copulas according to a regular vine. In Bedford and Cooke (2002), the following representation theorem for the joint density as the product of (conditional) copulas assigned to the nodes of a regular vine on n elements and the marginal densities is proved (arguments of the functions have been suppressed to simplify notation):

Theorem 2.1. *Let (F, V, B) be a copula vine specification where: $F = (F_1, \ldots, F_n)$ and each F_i has density f_i, $i = 1, \ldots, n$, $V(n) = (T_1, \ldots, T_{n-1})$ is a regular vine on n elements and $B = \{C_{jk|D_e} \mid e(j,k) \in \bigcup_{i=1}^{n-1} E_i$, where $e(j,k)$ is the unique edge with conditioned set $\{j,k\}$, and $C_{jk|D_e}$ is a copula for $\{X_j, X_k\}$ conditional on X_{D_e} with density $c_{jk|D_e}\}$. Then the vine-dependent distribution for (F, V, B) is uniquely determined and has a density given by*

$$f_{1,\ldots,n} = f_1 \cdots f_n \prod_{i=1}^{n-1} \prod_{e(j,k)\in E_i} c_{jk|D_e}(C_{j|D_e}, C_{k|D_e}).$$

where

$$C_{i|A} = \frac{\partial C_{ij|A\setminus\{j\}}(C_{i|A\setminus\{j\}}, C_{j|A\setminus\{j\}})}{\partial C_{j|A\setminus\{j\}}}$$

Hence based on $V_1(5)$ the density decomposition can be

$$f_{12345} = f_5 c_{35} f_3 c_{25|3} c_{23} f_2 c_{45|23} c_{42|3} c_{43} f_4 c_{15|234} c_{14|23} c_{13|2} c_{12} f_1.$$

For the m-vine $V(\{1,2,3\},\{2,3,4\},\{3,5\})$, obtained by removing nodes in the dashed area in $V_1(5)$, the density decomposition can be represented as follows,

$$f_{12345} = f_5 c_{35} f_3 c_{23} f_2 c_{42|3} c_{43} f_4 c_{13|2} c_{12} f_1.$$

Comparing with the decomposition based on $V_1(5)$, the copulas $c_{15|234}, c_{14|23}, c_{45|23}$ and $c_{25|3}$ are set to be independent copula.

The estimation of parameters in vine copulas is performed tree by tree which is efficient. Moreover, an efficient sampling procedure for vine copula models exists as well (Aas et al., 2009; Czado, 2019). For independent standard uniformly distributed variables $U_i, i = 1, \ldots, 5$, one possible sampling procedure for the m-vine $V(\{1,2,3\},\{2,3,4\},\{3,5\})$ obtained by removing nodes in the dashed area in $V_1(5)$ is as follows,

$$\begin{aligned}
X_5 &= F_5^{-1}(U_5),\\
X_3 &= F_3^{-1}\left(C_{3|5}^{-1}(U_3|U_5)\right),\\
X_2 &= F_2^{-1}\left(C_{2|3}^{-1}(U_2|U_3)\right),\\
X_4 &= F_4^{-1}\left(C_{4|3}^{-1}\left(C_{4|23}^{-1}(U_4|C_{2|3}^{-1}(U_2|U_3))|U_3\right)\right),\\
X_1 &= F_1^{-1}\left(C_{1|2}^{-1}\left(C_{1|23}^{-1}(U_1|C_{2|3}^{-1}(U_2|U_3))|U_2\right)\right)
\end{aligned}$$

By taking into account the conditional independence, one is able to simplify the estimation and also the sampling procedure for the vine copula model. We will show in the next section that an m-vine can be constructed when the pattern of conditional independence assumed in the model corresponds to the pattern of a strongly chordal graph.

3. Relationship between m-vine and strongly chordal graph

In Kurowicka and Cooke (2006), Hobæk Haff et al. (2016), it has been shown that there exists a relationship between chordal graphs and m-vines (or truncated vines). We discuss it briefly in Section 3.1 and conclude that m-vines correspond to chordal graphs, but not all chordal graphs correspond to m-vines (this has been shown in Hobæk Haff et al. (2016) for truncated vines). A more general discussion on this topic follows, leading to the main result of this paper and its proof. We show the equivalence of m-vines and strongly chordal graphs.

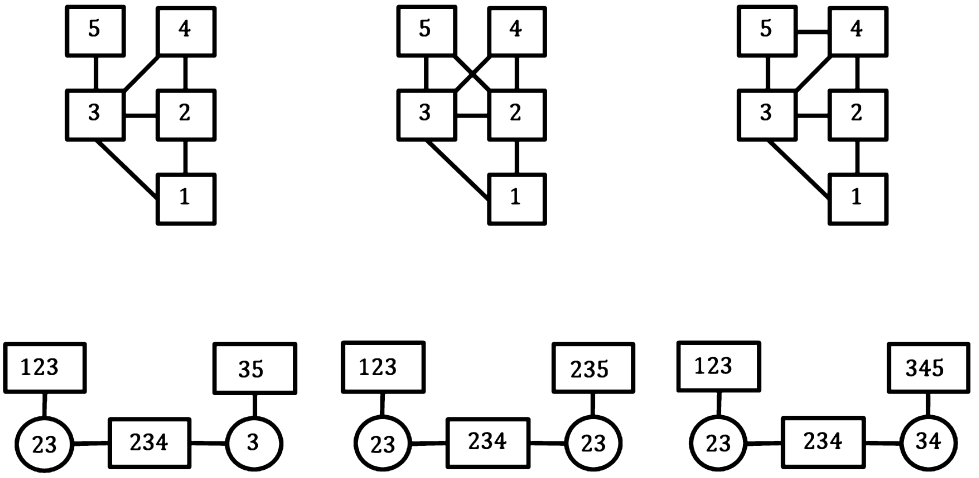

Fig. 5. Examples of clique trees (below) of chordal graphs (above). The graph on the left corresponds to the m-vines in Fig. 2 where the nodes in the dashed area have been removed. The middle and the right graphs correspond to the 3-truncated vines for $V_1(5)$ and $V_2(5)$ in Fig. 2, respectively.

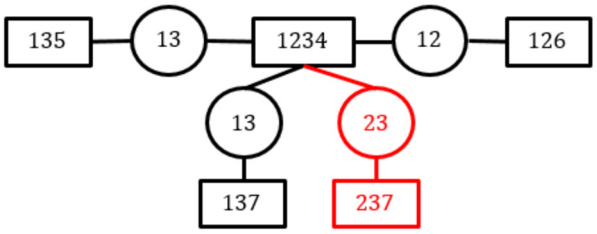

Fig. 6. A regular (black, $\mathcal{T}$) and non-regular (red, $\mathcal{T}'$) clique tree. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

3.1. Chordal graph and m-vine

One of the many characterizations of chordal graphs is the existence of a corresponding clique tree (Gavril, 1974). To show that an m-vine corresponds to a chordal graph, it is sufficient to observe that constraint sets of maximal nodes in an m-vine form maximal cliques and the separators are the intersections of constraint sets of adjacent maximal nodes. A tree obtained in this way is a clique tree of a chordal graph (Kurowicka and Cooke, 2006; Hobæk Haff et al., 2016).

As an example, let us consider the m-vine obtained by removing nodes in dashed areas shown in Fig. 2. For both m-vines, the corresponding clique tree (maximal cliques shown in squares and separators are in circles) and the chordal graphs are shown in Fig. 5 (left panel - the chordal graph for the m-vines when the nodes in the dashed area are removed for both vines; middle panel - the chordal graph for the 3-truncated vine in Fig. 2 (left); right panel - the chordal graph for the 3-truncated vine in Fig. 2 (right)).

It is not, however, the case that all chordal graphs correspond to m-vines. We call those clique trees to which a corresponding m-vine exists as **regular clique trees**.

Definition 3.1 *(Regular clique tree).* A clique tree $\mathcal{T}$ is regular if there exists an m-vine $V(U^*_{e_1}, \ldots, U^*_{e_k})$ such that the constraint sets of the maximal nodes $e_1, \ldots, e_k$ of the m-vine are the nodes $\mathcal{V}(\mathcal{T}) = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ of the clique tree, respectively.

In Fig. 6, we show an example of a regular and a non-regular clique tree. The 'black' tree is a regular clique tree and the non-regular clique tree is obtained when node $\{1, 3, 7\}$ is replaced by $\{2, 3, 7\}$, and we print it in red.

For the 'black' clique tree $\mathcal{T}$ in Fig. 6, there are four cliques shown in squares and three separators shown in circles. In this example, the corresponding m-vine can be constructed as follows: the clique with cardinality (number of elements) equal to 4 is the constraint set of the maximal node in tree T_4 of the m-vine, whereas the cliques with cardinality equal to 3 are the constraint sets of maximal nodes in T_3. The node with constraint set $\{1, 2, 3, 4\}$ has two children in T_3 and they share the common child 13 with node $\{1, 3, 5\}$ and $\{1, 3, 7\}$. Moreover, they share the common child 12 with node $\{1, 2, 6\}$. One possible m-vine can then have maximal nodes 26|1, 15|3, 17|3 and 23|14 (it has two children 24|1 and 34|1).

When we look at the 'red' clique tree $\mathcal{T}'$, where node $\{1, 3, 7\}$ is replaced by $\{2, 3, 7\}$, following the arguments above, we can observe that the children of node $\{1, 2, 3, 4\}$ should share the common node 13 with node $\{1, 3, 5\}$, the common

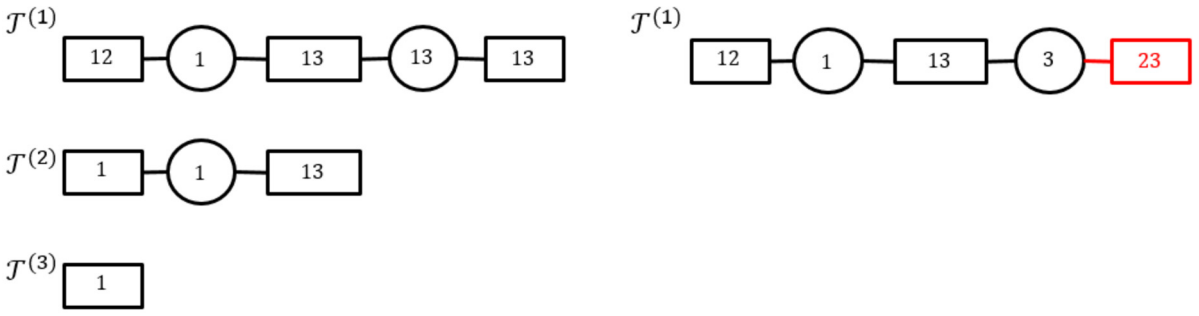

Fig. 7. Sequence of trees $\mathcal{T}^{(i)}$, $i = 1, \ldots$ for the clique trees in Fig. 6 (black (left) and red (right)), respectively.

node 12 with node $\{1, 2, 6\}$ and the common node 23 with node $\{2, 3, 7\}$. This is not possible for a regular vine as 13, 12 and 23 constitutes a cycle.

3.2. Strongly chordal graph and m-vine

As proved in McKee (2003), strongly chordal graphs are characterized by the existence of a strong clique tree, defined below:

Definition 3.2 *(Strong clique tree).* Clique tree $\mathcal{T}$ is a strong clique tree of graph $\mathcal{G}$ if there exists the sequence of trees $\mathcal{T}^{(0)}, \mathcal{T}^{(1)}, ..., \mathcal{T}^{(k)}$, where $\mathcal{T}^{(0)} = \mathcal{T}$, $\mathcal{T}^{(k)}$ is edgeless and $\mathcal{T}^{(i)}$ is a maximum spanning tree of the separators $\boldsymbol{\mathcal{S}}(\mathcal{T}^{(i-1)})$ with edge weight being the cardinality of separator such that $\mathcal{T}^{(i)}_{\{v\}}$ is connected for any vertex v in $\mathcal{G}$.

In Fig. 7, the sequence of trees $\mathcal{T}^{(i)}, i = 1, \ldots$ for clique trees in Fig. 6 are shown. We can observe that in $\mathcal{T}^{(1)}$ (right) node $(1, 2)$ and $(2, 3)$ are not connected.

In McKee (2003) another characterization of a strong clique tree was proposed.

Theorem 3.1. *A clique tree $\mathcal{T}$ of a graph $\mathcal{G}$ is strong if and only if there do not exist distinct vertices $v_1, ..., v_k \in \mathcal{V}(\mathcal{G})$ and distinct nodes $\mathcal{C}_1, \ldots, \mathcal{C}_k \in \mathcal{V}(\mathcal{T})$ where $k \geq 3$ and, for each i, $\mathcal{C}_i \cap \{v_1, ..., v_k\} = \{v_i, v_{i+1}\}$ (computing subscripts modulo k).*

Using the above theorem, we see immediately that the red clique tree in Fig. 6 is not strong as for cliques $\mathcal{C}_1 = \{1, 3, 5\}$, $\mathcal{C}_2 = \{1, 2, 6\}$, $\mathcal{C}_3 = \{2, 3, 7\}$ and a set of vertices $\{v_1 = 3, v_2 = 1, v_3 = 2\}$ the condition in the theorem is violated.

In the remaining part of this section, we prove that a strong clique tree corresponds to an m-vine. We only consider the situation when the strong clique tree is connected, otherwise the proof can be applied to each connected subgraph of the strong clique tree, separately.

Theorem 3.2. *Clique tree $\mathcal{T}$ is regular if and only if $\mathcal{T}$ is strong.*

Proof. Necessity: Assume that $\mathcal{T}$ is regular but $\mathcal{T}$ is not strong. By Theorem 3.1, there exist distinct vertices $v_1, \ldots, v_k \in \mathcal{V}(\mathcal{G})$ and distinct nodes $\mathcal{C}_1, \ldots, \mathcal{C}_k \in \mathcal{V}(\mathcal{T})$ where $k \geq 3$ and, for each i, $\mathcal{C}_i \cap \{v_1, ..., v_k\} = \{v_i, v_{i+1}\}$ (Note that $\mathcal{C}_k \cap \{v_1, ..., v_k\} = \{v_k, v_1\}$). Each $\mathcal{C}_i$ corresponds to the constraint set of a maximal node in the m-vine (a subvine of this m-vine), and they intersect by forming a path through vertices $\{v_1, v_2, \ldots, v_k, v_1\}$. This leads to a cycle in the first tree of m-vine which is not allowed by the definition of a vine.

Sufficiency: The proof of sufficiency is by construction and is presented in Section 3.4. □

Before illustrating the algorithm to prove sufficiency, we show in the next section the algorithm of merging vines. This lays the foundation for the algorithm of constructing an m-vine corresponding to a strongly chordal graph.

3.3. Merging vines

In this section, an algorithm to combine two regular vines $V(n)$ and $V(m)$ into a larger vine, such that $V(n)$ and $V(m)$ are its subvines is presented. This algorithm lays the foundation for the procedure to construct an m-vine corresponding to a strongly chordal graph.

We call the procedure of combining two vines **merging**. This procedure has been introduced in Cooke et al. (2015) for vines without overlapping elements. In that case merging two vines is always possible. If we want to combine two vines that share the same subset of elements, then merging is possible only if they share a common subvine on common elements. A formal definition is as follows.

Definition 3.3 *(Merger).* A regular vine $V(\mathcal{V}_n \cup \mathcal{V}_m)$ is a merger of two regular vines $V(n)$ and $V(m)$ where $\mathcal{V}_n \nsubseteq \mathcal{V}_m$ and $\mathcal{V}_m \nsubseteq \mathcal{V}_n$ if $V(n)$ and $V(m)$ are subvines of $V(\mathcal{V}_n \cup \mathcal{V}_m)$.

Construction of the merger for two vines $V(n)$ and $V(m)$ with common subvine $V(p)$ can be done following a bottom-up approach, where each tree structure starting from tree T_{p+1} in the merger is constructed sequentially. This can be achieved following Algorithm 1.

Algorithm 1 General approach of merging two regular vines $V(n)$ and $V(m)$.

Input: regular vine $V(n)$ and $V(m)$ where $\mathcal{V}_n \cap \mathcal{V}_m = \mathcal{V}_p$ (possibly empty)
Output: A merger $V(\mathcal{V}_n \cup \mathcal{V}_m)$
1: **for** i in $p+1$ to $n+m-p$ **do**
2: Construct the tree structure T_i of the merger satisfying three conditions:
- the proximity condition;
- if $i \leq \max(n, m)$, nodes in tree T_i of the merger should include the nodes in T_i of $V(n)$ and $V(m)$;
- if $i \leq \max(n, m)$, connections between nodes of the merger have to be consistent with connections that these nodes have in $V(n)$ and $V(m)$.

3: **end for**

In Cooke et al. (2015), the merging algorithm has been presented in the case of two vines that do not overlap, hence when $\mathcal{V}_n \cap \mathcal{V}_m = \mathcal{V}_p = \emptyset$. The possible merger of two vines without overlap was shown to be determined by the choice of **sampling order** of $V(n)$ and $V(m)$. This gives a more efficient way of constructing the merger when compared to the method in Algorithm 1, and allows to prove that there are always 2^{n+m-2} possible mergers of $V(n)$ and $V(m)$. The procedure developed to merge vines without overlap can be adapted to work also in the case of vines with overlap. Hence we first present a short explanation of the sampling order (the formal definition can be found in Cooke et al. (2015)) and then discuss the merging.

The sampling order is an order of elements that can be found as follows: according to the property of regular vines, an element in the conditioned set of the top node is partnered with exactly one of the remaining elements in each tree (is in the conditioned set of exactly one node in each tree). These nodes are easily removed one by one from the regular vine. We call the process of removing these nodes **marginalization**. More precisely, a marginalization of a regular vine $V(n)$ by element v_i, where v_i is in the conditioned set of the top node of $V(n)$, is to remove all nodes whose conditioned set contains v_i (one in each tree). After marginalization, one obtains the subvine of $V(n)$, $V(\mathcal{V}_n \setminus \{v_i\})$. This subvine can be further marginalized by an element from its conditioned set. Applying marginalization successively, we finally get just one element. The order of elements used in this successive process corresponds to the reverse sampling order.

Suppose we have a sampling order $(a_1, a_2, \ldots, a_n)$ for $V(n)$ and a sampling order $(b_1, b_2, \ldots, b_m)$ for $V(m)$, the new partners of a_n in nodes from T_{n+m} to T_{n+1} in the merger are the elements in reverse sampling order, $(b_m, \ldots, b_1)$. After marginalizing element a_n from the merger, the new partners of a_{n-1} in nodes from T_{n-1+m} to T_n in the merger will be the elements $(b_m, \ldots, b_1)$. In general, the new partners of a_i will be chosen following the reverse sampling order, $(b_m, \ldots, b_1)$, from T_{i+m} to T_{i+1}. Similarly the partners of b_j will be $(a_n, \ldots, a_1)$ from T_{j+n} to T_{j+1} in the merger. A general algorithm is as follows (which appears in the proof of Theorem 5.2 in Cooke et al., 2015),

Algorithm 2 Merging two regular vines $V(n)$ and $V(m)$ without overlapping elements.

Input: regular vine $V(n)$ and $V(m)$ where $\mathcal{V}_n \cap \mathcal{V}_m = \emptyset$
Output: A merger $V(\mathcal{V}_n \cup \mathcal{V}_m)$
1: Determine a sampling order $SO_n = (a_1, \ldots, a_n)$ for $V(n)$.
2: Determine a sampling order $SO_m = (b_1, \ldots, b_m)$ for $V(m)$.
3: The merger is constructed based on SO_n and SO_m where the new partners of a_i from T_{i+m} to T_{i+1} will be $(b_m, \ldots, b_1)$, and the new partners of b_j from T_{j+n} to T_{j+1} will be $(a_n, \ldots, a_1)$.

The example to illustrate Algorithm 2 is presented in Appendix A.1. Extending a regular vine by just one distinct element, introduced in Nápoles (2010), is a special case of merging two vines without overlap. In this case, one vine is just a single element and its sampling order is this element itself.

In the case when $V(n)$ and $V(m)$ have a common subvine $V(p)$, $\mathcal{V}_p \neq \emptyset$, the tree structures up to and including T_p of the merger are fixed, hence only the segment $(a_{p+1}, \ldots, a_n)$ and $(b_{p+1}, \ldots, b_m)$ will contribute to the construction of the merger when Algorithm 2 is applied. However, not all sampling orders lead to a valid merger. One extra constraint that both the initial segments $(a_1, \ldots, a_p)$ and $(b_1, \ldots, b_p)$ are sampling orders of the common subvine $V(p)$ should be satisfied (a detailed proof can be seen in Appendix A.2). This constraint guarantees that the construction of the merger from T_{n+m-p} to T_{p+2} follows a vine structure construction for merger without overlap, and a valid tree structure construction for T_{p+1}, hence a valid merger can be constructed according to Algorithm 2.

As an example, we show how to merge vine $V_1(5) = V(\{1, 2, 3, 4, 5\})$ and another vine denoted as $V'(5) = V(\{2, 3, 6, 7, 8\})$, which share a common subvine on elements $\{2, 3\}$. These two vines are shown in Fig. 8 (black).

Following Algorithm 2 with extra conditions on the sampling orders, one possible merger of $V_1(5)$ and $V'(5)$ is shown in Fig. 8, where the sampling order is $(2, 3, 5, 4, 1)$ or $(3, 2, 5, 4, 1)$ in $V_1(5)$ and $(2, 3, 6, 8, 7)$ or $(3, 2, 6, 8, 7)$ in $V'(5)$. Other

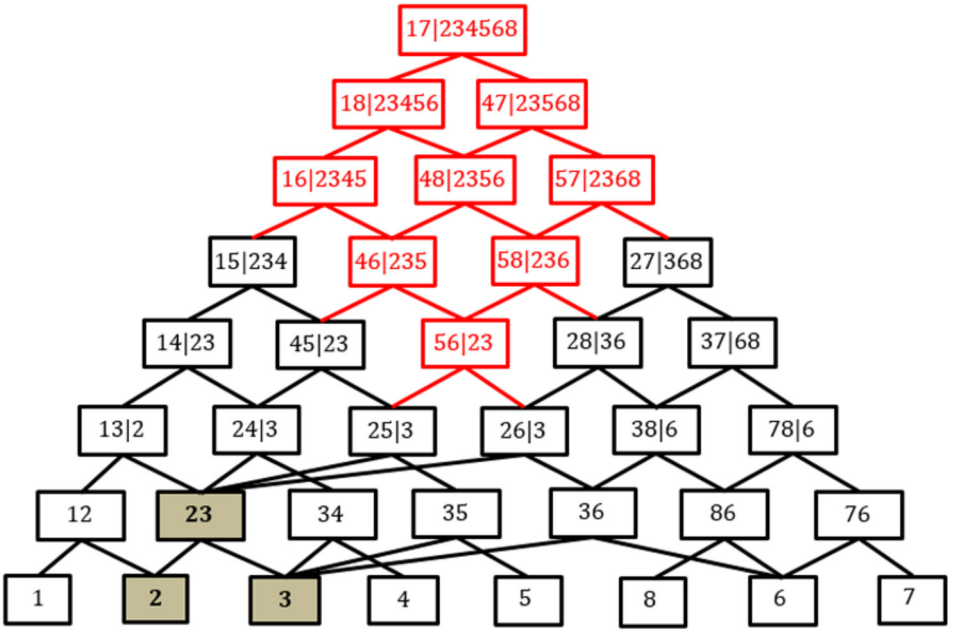

Fig. 8. A merger of vine $V_1(5)$ ($V(\{1,2,3,4,5\})$) (black) and $V'(5)$ ($V(\{2,3,6,7,8\})$) (black). The nodes in the common subvine $V(\{2,3\})$ is denoted in shaded area and the new nodes and lines in the vine triangular array are marked in red.

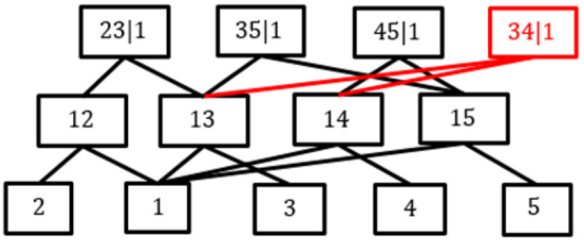

Fig. 9. Three regular vines $V(\{1,2,3\})$ ($V_1(3)$), $V(\{1,3,5\})$ ($V_2(3)$) and $V(\{1,4,5\})$ ($V_3(3)$) (black) and one possible way of merging $V_1(3)$ and $V_3(3)$ (red).

choice for the segment (a_3, a_4, a_5) in $V_1(5)$ can be $(4,5,1)$, $(4,1,5)$ or $(1,4,5)$. There is no other choice for the segment (b_3, b_4, b_5) in $V'(5)$. Hence in total there are four possible mergers of $V_1(5)$ and $V'(5)$.

In general it is unknown how many mergers can be constructed. The number of mergers depends on the structures of both vines and the overlapping elements. In this paper, we follow the bottom-up approach in Algorithm 1 when we want to merge two vines with overlap (we expect differences in the performance of both algorithms but this problem will not be researched in this paper). The merger given in Fig. 8 is constructed by connecting nodes 25|3 and 26|3 in T_3. Other choice can be to connect 13|2 and 26|3 (leading to another merger) or to connect 24|3 with 26|3 (leading to two extra mergers).

In the examples above we have explained how two vines can be merged. However, when there are more overlapping vines that need to be merged, the order of merging should be specified so the merging is done consistently. We show in Fig. 9 one example where three vines $V(\{1,2,3\})$, $V(\{1,3,5\})$ and $V(\{1,4,5\})$ (denoted as $V_1(3)$, $V_2(3)$ and $V_3(3)$, respectively) are to be merged.

If we were to merge $V_1(3)$ and $V_3(3)$ first, setting the new node in T_3 of $V\{1,2,3,4,5\}$ to be 34|1 (in red) instead of 35|1, as shown in Fig. 9, then their merger will not have a common subvine with $V_2(3)$. By choosing right a correct order of merging these vines the problem above can be avoided. In the example above, we should merge $V_1(3)$ and $V_2(3)$ first and then $V_3(3)$, or to merge $V_2(3)$ and $V_3(3)$ first and then $V_1(3)$. In Section 3.4, we will provide a procedure to order the merging process when an m-vine corresponding to a strong clique tree is constructed.

3.4. Construction of an m-vine corresponding to a strong clique tree

In this section, an algorithm to construct an m-vine corresponding to a strong clique tree is presented. Regular vines on elements of each node in the strong clique tree $\mathcal{T}$ are constructed and they are constructed consistently such that they contain common subvines. This is done by following, in reverse order, the sequence of trees $\mathcal{T}^{(i)}, i = 0, \ldots, k$ obtained for $\mathcal{T}$. The nodes of these trees determine constraint sets of nodes that have to appear at different trees in a regular vine. The

elements in a node in $\mathcal{T}^{(i)}$ are either in its separator(s) or do not appear in any node of $\mathcal{T}^{(j)}$, $j \geq i$, which we refer to as **extra elements**.

First, an example of the approach will be presented. The m-vine corresponding to the strong clique tree $\mathcal{T}$ in Fig. 6 (black) can be constructed as follows:

The sequence of trees $\mathcal{T}^{(i)}, i = 1, \ldots, 3$ are in Fig. 7 (left). We follow these trees in reverse order starting with $\mathcal{T}^{(3)}$.

Step 1: In $\mathcal{T}^{(3)}$ there is only one node, which has no separator (the process is just started) and has an extra element 1. Hence, a vine $V(\{1\})$ is constructed.

Step 2: In $\mathcal{T}^{(2)}$ we have $\mathcal{C}_1^{(2)} = \{1\}$ and $\mathcal{C}_2^{(2)} = \{1, 3\}$. Node $\mathcal{C}_1^{(2)}$ has a separator $\{1\}$ and no extra element, hence the vine is $V(\{1\})$. Node $\mathcal{C}_2^{(2)}$ has a separator $\{1\}$ and an extra element 3, then the vine for this node is obtained through extending $V(\{1\})$ by element 3, to get $V(\{1, 3\})$.

Step 3: In $\mathcal{T}^{(1)}$, there are three nodes $\mathcal{C}_1^{(1)} = \{1, 2\}, \mathcal{C}_2^{(1)} = \{1, 3\}$ and $\mathcal{C}_3^{1} = \{1, 3\}$. Node $\mathcal{C}_1^{(1)}$ has separator $\{1\}$ and an extra element 2, hence we obtain $V(\{1, 2\})$ by extending $V(\{1\})$ with element 2. In the case of node $\mathcal{C}_2^{(1)}$, we see that it has separators $\{1\}$ and $\{1, 3\}$. Separator $\{1\}$ is a subset of $\{1, 3\}$, hence only separator $\{1, 3\}$ will be considered. Since there is no extra element its corresponding vine is $V(\{1, 3\})$. Node $\mathcal{C}_3^{(1)}$ has separator $\{1, 3\}$ and no extra element so its corresponding vine is $V(\{1, 3\})$.

Step 4: In $\mathcal{T}^{(0)}$ corresponding vines for nodes $\{1, 3, 5\}$, $\{1, 2, 6\}$ and $\{1, 3, 7\}$ can be constructed similarly as above. Node $\{1, 2, 3, 4\}$ has two separators, $\{1, 2\}$ and $\{1, 3\}$, with their corresponding vines $V(\{1, 2\})$ and $V(\{1, 3\})$, respectively, and an extra element 4. A vine $V(\{1, 2, 3\})$ can be constructed by at first merging $V(\{1, 2\})$ and $V(\{1, 3\})$ and then by extending with element 4 to get $V(\{1, 2, 3, 4\})$. The last step is to combine the regular vines in $\mathcal{T}^{(0)}$ to get the m-vine $V(\{1, 2, 6\}, \{1, 3, 5\}, \{1, 2, 3, 4\}, \{1, 3, 7\})$.

The general algorithm to construct an m-vine corresponding to a strong clique tree $\mathcal{T}$ is presented in Algorithm 3.

Algorithm 3 Construction of an m-vine for a strong clique tree $\mathcal{T}$.

Input: A tree sequence $\mathcal{T}^{(i)}, i = 0, \ldots, k$ where $\mathcal{T}^{(0)} = \mathcal{T}$
Output: An m-vine corresponding to the strong clique tree $\mathcal{T}$

```
1: for i from k to 0 do
2:     repeat
3:         Find the separators of C_j^(i) which are not subsets of the others.
4:         if no separator found then
5:             Construct a regular vine V(C_j^(i)) for the node C_j^(i).
6:         else
7:             Find the vine structures corresponding to the separators.
8:             Combine the vines corresponding to separators and the extra elements (if applicable).
9:             The vine after merging and extending by extra elements is V(C_j^(i)).
10:        end if
11:    until all nodes in T^(i) have been considered
12: end for
13: Combine the vine structures corresponding to the nodes in T^(0) to get the m-vine.
```

As discussed in the end of Section 3.3, a correct order of merging more than two regular vines is needed. This can be implemented in line 8 in Algorithm 3 where more than two separators of node $\mathcal{C}_j^{(i)}$ are found. These separators are nodes in tree $\mathcal{T}^{(i+1)}$. We propose to merge them following the known depth-first search algorithm of trees. That is, to choose randomly one separator as a root of the tree and then order separators by applying a depth-first algorithm in $\mathcal{T}^{(i+1)}$. The detailed algorithm can be found in Algorithm 5 in the Appendix A.5. Due to the property of strong clique tree, the merging of the separators this way is always consistent.

Theorem 3.3 below ensures that Algorithm 3 produces a consistent m-vine. We show that subvines corresponding to each node in $\mathcal{T}^{(i)}, i = 0, \ldots, k$ are constructed such that if they have overlapping elements they must share a common subvine on the overlap.

Theorem 3.3. *Let $\mathcal{C}_m$ and $\mathcal{C}_n$ be nodes in the tree sequence $\mathcal{T}^{(i)}, i = 0, \ldots, k$ and let $\mathcal{C}_p = \mathcal{C}_m \cap \mathcal{C}_n \neq \emptyset$. Regular vines $V(\mathcal{C}_m)$ and $V(\mathcal{C}_n)$ constructed according to Algorithm 3 share a common subvine $V(\mathcal{C}_p)$.*

Proof. Suppose nodes $\mathcal{C}_m$ and $\mathcal{C}_n$ are in tree $\mathcal{T}^{(i_m)}$ and $\mathcal{T}^{(i_n)}$ respectively, where $i_m \leq i_n$. It suffices to find a node with elements $\mathcal{C}_p$ in the tree sequence. If $i_m = i_n$, then there is a path through nodes $\mathcal{C}_m$ and $\mathcal{C}_n$ and the overlapping elements $\mathcal{C}_p$ appear in at least one separator of $\mathcal{C}_m$ or $\mathcal{C}_n$ (if the separator is $\mathcal{C}_p$ then node is found). These separators are the nodes in tree $\mathcal{T}^{(i_m+1)}$ and there must be a path through these separators. Similarly, their separators containing elements $\mathcal{C}_p$ are the nodes in $\mathcal{T}^{(i_m+2)}$ and a path through them exists as well. Applying this argument iteratively, we can finally find a node in a certain tree level containing only elements $\mathcal{C}_p$. The common subvine $V(\mathcal{C}_p)$ of $V(\mathcal{C}_m)$ and $V(\mathcal{C}_n)$ is built at that stage

Table 1
An m-vine $V(\{2\},\{6\},\{1,3,7,8\},\{1,4,5,9,10\})$.

Structure	C	τ	Structure	C	τ	Structure	C	τ
Tree-4								
$9,5\|1,4,10$	C270	-0.60						
Tree-3								
$5,10\|1,4$	C180	0.48	$1,7\|3,8$	G270	-0.34	$9,10\|1,4$	C270	-0.67
Tree-2								
$4,5\|1$	G180	0.74	$3,7\|8$	C90	-0.68	$1,9\|4$	C90	-0.50
$1,3\|8$	C90	-0.70	$1,10\|4$	G270	-0.88			
Tree-1								
$1,5$	G270	-0.33	$7,8$	C	0.61	$4,9$	G180	0.49
$3,8$	G	0.50	$4,10$	G	0.54	$1,4$	G180	0.56
$1,8$	C180	0.64						

and will not change anymore during the construction. When $i_m < i_n$, the search procedure starts with the separator(s) of $\mathcal{C}_m$ that contains elements $\mathcal{C}_p$. This separator will finally result in node(s) that includes the elements $\mathcal{C}_p$ in $\mathcal{T}^{(i_n)}$, otherwise the property that $\mathcal{T}^{(i)}_{\{v\}}$ is connected is violated. Similarly, as in the case when $i_m = i_n$, a node $\mathcal{C}_p$ and the corresponding common subvine $V(\mathcal{C}_p)$ can be found. □

In the next section we show how the theoretical results presented above can be applied in practice.

4. Applications of the equivalence between m-vines and strongly chordal graphs

The theoretical result presented in the previous section provides a flexible method that estimates a distribution with strongly chordal pattern of conditional independence. Using our result one construct a variety of distributions having a specified set of conditional independence and different properties, e.g. correlation structures, asymmetries, tail dependence. This can be valuable, e.g. in simulation studies that evaluate the performance of conditional independence tests in a non-Gaussian environment. Moreover one can take an advantage of existing software implementations of vine copula models.

However, in this section we concentrate on a different application of our result. It is known that the number of regular vine structures on n elements grows exponentially with dimensions. Hence estimating all vine structures for the given data is infeasible. Simplifying the model by assuming a pattern of conditional independence to be presented in the data, represented by a strongly chordal graph, makes the estimation effort manageable. The number of m-vine structures constructed following Algorithm 3 is much smaller (however the exact number of m-vine structures is unknown). For clique trees containing a small enough number of elements in maximal nodes, one can assess all vine structures for the subvines corresponding to the maximal nodes. Furthermore one can even estimate a non-simplified regular vine.

In this section, we first analyze in detail two simulated examples where the advantages of simplifying the vine copula model are highlighted. In high dimensions when it is not possible to estimate all possible m-vines, a heuristic method to choose an m-vine structure for a given data is needed. We propose a general heuristic of constructing m-vine structures from data, along with a simulation study to evaluate its performance.

The following procedure is followed to obtain a strongly chordal graph from data. For a given data transformed into standard normal scale (data with uniformly distributed margins simulated from the m-vine is transformed by applying the inverse standard normal cumulative distribution function to each margin), the strongly chordal graph is constructed as follows: a graph is chosen at first using function `selectFast` in the **GGMselect** package (Giraud et al., 2009) (the best graph in a family of graphs generated by the method introduced in Meinshausen and Bühlmann, 2006; Wille and Bühlmann, 2006 is chosen), if this graph is not chordal, necessary edges are added to make it chordal; if this graph is not strongly chordal (according to Theorem 3.1), more edges are added based on the algorithm in Mukhopadhyay and Rahman (2021).

4.1. Simulated examples

An m-vine on 10 elements is used in simulations. The maximal nodes are $\{2\}$, $\{6\}$, $\{1,3,7,8\}$ and $\{1,4,5,9,10\}$. Details about the simulated vine are presented in Table 1 (in this example, we include commas to separate elements in the nodes of the vine and in the nodes of the clique tree), where we show the m-vine structure, the copula family C and the corresponding Kendall's tau τ. In order to amplify the effect of the vine structure, the copula families are chosen from Clayton copula (C), Gumbel copula (G) and their rotated versions. The vine triangular array for this m-vine is shown in Appendix A.3.

4.1.1. Ability to assess all possible vine structures

The number of vines grows exponentially as dimension grows. However in the m-vine approach, we may be able to find the best vine structure for a maximal clique with relatively small number of variables. We show below an example.

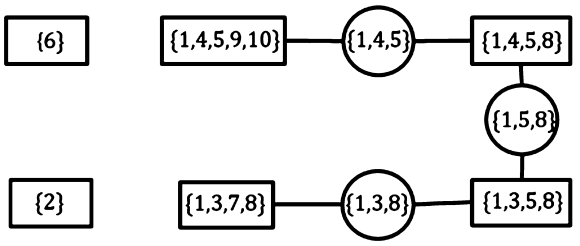

Fig. 10. The strong clique tree estimated by `selectFast` for the data simulated from the m-vine with 1000 samples.

Table 2
The AIC, BIC and the p-value from a Vuong test for the true m-vine (true), the m-vine we constructed (m-vine) corresponding to the strong clique tree in Fig. 10 and a 5-truncated vine constructed by Dißmann's heuristic (5-trun). The upper diagonal in the Vuong test shows the result without Schwarz correction while the lower diagonal shows the result with Schwarz correction.

	AIC	BIC	Vuong test p-value		
			true	m-vine	5-trun
true	−20438.98	−20360.46		2.83E−15	3.02E−117
m-vine	−20036.67	−19913.97	4.48E−20		1.77E−110
5-trun	−18134.05	−18016.27	7.03E−123	1.08E−109	

We simulate 1000 samples from the m-vine. The strong clique tree from the data is as follows.

The graph given by the function `selectFast` is already a strongly chordal graph and we observe that it does not exactly recover the clique tree of the true m-vine. (This is not surprising as the methods GGMselect assumes gaussian data.) Extra maximal cliques appear in the estimated tree, $\{1,4,5,8\}$ and $\{1,3,5,8\}$. However, we show below that one can still find the true structure (copula families can be different) for subvine $V(\{1,4,5,9,10\})$ and $V(\{1,3,7,8\})$.

According to Algorithm 3, $V(\{1,4,5,9,10\})$ is constructed by extending $V(\{1,4,5\})$ with variable 9 and 10, and $V(\{1,3,7,8\})$ is constructed by extending $V(\{1,3,8\})$ with variable 7. Vine $V(\{1,4,5\})$ requires to contain node 1,5, hence only two possible structures are available. The best structure is the true structure which has the smallest $AIC = -3207.56$ (also smallest $BIC = -3183.03$). When we fix the best structure for $V(\{1,4,5\})$, there are 48 possible vine structures for $V(\{1,4,5,9,10\})$ and it is easy to estimate all of them. The one that has the smallest $AIC = -12394.17$ (also smallest $BIC = -12330.37$) is the true structure for these variables. Similarly, for $V(\{1,3,7,8\})$, its subvine $V(\{1,3,8\})$ has two possible structures with node 1,8 included. The best $V(\{1,3,8\})$ is the one we simulated from and has $AIC = -4202.58$ ($BIC = -4182.95$). Then extending $V(\{1,3,8\})$ by variable 7 leads to estimating four possible structures and the best one is the true one $V(\{1,3,7,8\})$ ($AIC = -7638.84$ and $BIC = -7599.58$).

The remaining maximal cliques $\{1,4,5,8\}$ and $\{1,3,7,8\}$ are determined once $V(\{1,4,5\})$ and $V(\{1,3,8\})$ are fixed (In general finding the best vine for each maximal clique does not guarantee a globally optimal m-vine). We compare the m-vine estimated following the procedure above with the true m-vine and observe that the contribution to AIC and BIC of the extra nodes $5,8|1,\ 3,5|1,8$ and $4,8|1,5$ in the constructed m-vine is very small. Following Dißmann's heuristic in Dißmann et al. (2013); Brechmann et al. (2012), a 5-truncated vine is constructed. We see in Table 2 that the m-vine we constructed is significantly better than the truncated vine. However, even when the true structure is retrieved, due to estimation error from the tree-wise estimation procedure and the limited choice of parametric copula families, the constructed m-vine performs worse than the true m-vine for the data.

4.1.2. Ability to assess non-simplified models for subvines

Non-simplified vines are difficult to construct and estimate. There are a few attempts to handle this problem, which propose to estimate the relationship between the copula parameter (or the corresponding conditional Kendall's tau) and the conditioning variables non-parametrically (Acar et al., 2011) or to use generalized additive models (GAM) (Nagler and Vatter, 2020). A discussion about the performance of these two methods can be found in Acar et al. (2019). These approaches are in principle feasible only in low dimensions. In this paper, we show an example using the latter method which is implemented in the **gamCopula** package.

The non-simplified regular vine is constructed based on the same m-vine as in Table 1. We replace the constant conditional copula by assuming a relationship between the conditional Kendall's tau and the conditioning variables in the subvines $V(\{1,4,5\})$, $V(\{1,4,10\})$ and $V(\{1,4,9\})$ as follows: $\tau = \frac{e^z-1}{e^z+1}$, where $z = -\beta_0 + \beta_1 u$ and u denotes the conditioning variable. The copula families are adjusted to include their rotated versions (the copula family for node $4,5|1$ ($\beta_0 = -2.77, \beta_1 = 5.70$) includes G180 and G270, for node $1,10|4$ ($\beta_0 = -1.63, \beta_1 = 3.28$) includes G and G270 and for node $1,9|4$ ($\beta_0 = -1.27, \beta_1 = 2.34$) includes C180 and C90). Again 1000 samples are simulated from this m-vine and the resulting strong clique tree presented in Fig. 11.

Edges 1,9 and 1,10 are added in order to make the graph strongly chordal. We see that in the estimated tree nodes 5 and 10 as well as 4 and 10 are not connected. Hence the true structure for $V(\{1,4,5,9,10\})$ cannot be found. Similar, as

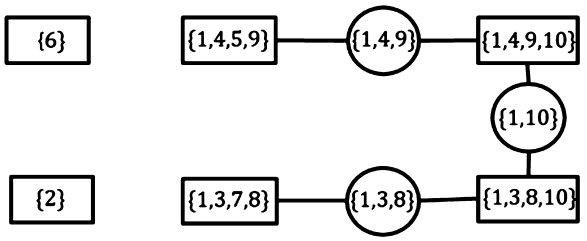

Fig. 11. The strong clique tree for the data simulated from a non simplified m-vine of size 1000 by adding edges 19 and 1, 10 to the estimated graph from `selecFast`.

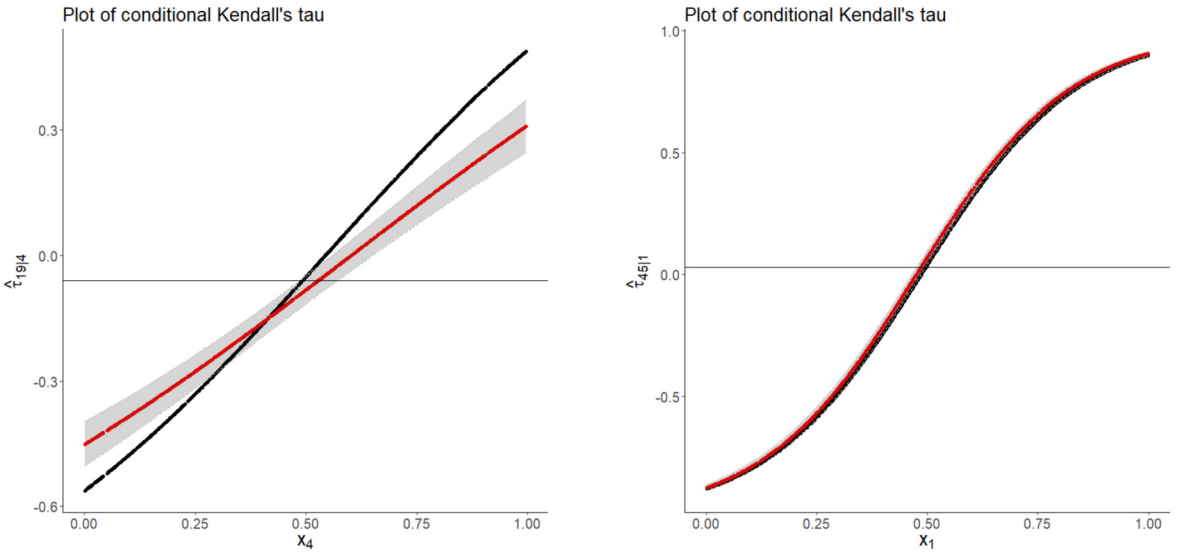

Fig. 12. Kendall's tau for the conditional copula in vine $V(\{1,4,9\})$ (left) and in vine $V(\{1,4,5\})$ (right). The true conditional tau is marked in black points and the estimated tau is marked in red points with its 95% confidence interval. The horizontal line denotes the conditional tau estimated by a simplified vine ($\tau = -0.06$ (left) and $\tau = 0.02$ (right)).

in the previous example, we can still find the best vine for each maximal clique. The maximal clique $\{1,4,5,9\}$ requires to construct $V(\{1,4,9\})$ first. There are three possible vine structures for $V(\{1,4,9\})$ and we first estimate a simplified vine for each structure. Then we apply the `pacotest` (Kurz, 2019) to check whether the simplifying assumption is rejected. If it is rejected, we fit a non-simplified vine with the GAM method instead. All the three structures are rejected to be a simplified vine, and the best non-simplified vine ($AIC = -1977.96$ and $BIC = -1953.42$) is the one with the true vine structure (the number of parameters in a non-simplified vine includes, 1) the number of parameters of the simplified bivariate (conditional) copulas, 2) the number of degrees of freedom in the GAM model contained in the non-simplified bivariate conditional copulas. 3) the number of the second parameters in the non-simplified bivariate conditional copulas if there are any). The relationship between the conditional tau and the conditioning variable is shown in Fig. 12 (left). The estimated non-simplified conditional copula is a Student-t copula with degrees of freedom equal to 15.33 and the coefficients are $\hat{\beta}_0 = -0.97, \hat{\beta}_1 = 1.62$.

Fixing the best non-simplified vine $V(\{1,4,9\})$ gives four possible structures for $V(\{1,4,5,9\})$. The best structure is to connect 5 and 1 in the first tree. A non-simplified vine is estimated which has $AIC = -3588.35$ and $BIC = -3517.44$. Furthermore, the non-simplified conditional copula $C_{45|1}$ can be estimated quite well, as shown in Fig. 12 (right). The vine structure for the maximal clique $\{1,4,9,10\}$ is fixed once $V(\{1,10\})$ (the **gamCopula** package does not allow the $BB6$ family copula ($AIC = -506.61$, $BIC = -496.79$) to be chosen which for this data has better AIC than the Gumbel copula ($AIC = -505.73$, $BIC = -500.82$). Hence we fix this copula to be Gumbel also in the later simplified vine $V(\{1,3,8,10\})$) is determined, which leads to a non-simplified vine $V(\{1,4,9,10\})$ with $AIC = -3116.49$ and $BIC = -3023.53$. The remaining two maximal cliques require to construct $V(\{1,3,8\})$ at first. Similarly there are three possible structures and for each structure a simplified vine is estimated or a non-simplified vine is estimated instead decided by the `pacotest`. Two of them are non-simplified vines but the best one is a simplified vine having the true structure ($AIC = -3908.47$ and $BIC = -3893.75$). Once $V(\{1,3,8\})$ is determined $V(\{1,3,8,10\})$ is fixed. $V(\{1,3,7,8\})$ is constructed by extending $V(\{1,3,8\})$ with variable 7 and there are four possibilities to consider. Two of them are rejected to be simplified vines hence non-simplified vines are estimated instead. The best $V(\{1,3,7,8\})$ is a simplified vine with its true structure ($AIC = -7139.28$ and $BIC = -7104.93$).

Table 3
The AIC, BIC and the p-value from a Vuong test for the true m-vine model (true), the m-vine we constructed (m-vine) corresponding to the strong clique tree in Fig. 11 and a 4-truncated vine constructed by Dißmann's heuristic (4-trun). The upper diagonal in the Vuong test shows the result without Schwarz correction while the lower diagonal shows the result with Schwarz correction.

	AIC	BIC	Vuong test p-value		
			true	m-vine	4-trun
true	−11865.60	−11772.35		0.775	2.02E−13
m-vine	−11865.28	−11688.48	0.494		1.64E−45
4-trun	−10812.19	−10714.04	1.40E−13	2.36E−37	

The Dißmann's heuristic gives a 4-truncated vine. Information about the performance of the models is provided in Table 3 and we see that the m-vine approach is significantly better.

Since we are not able to recover the true vine structure due to the restriction caused by the constructed strongly chordal graph, the dependence between X_1 and X_{10} conditional on X_4 cannot be estimated well, which is shown in Appendix A.4.

In the two simulated examples above, we have explained that by applying the m-vine approach we are able to focus on estimating its subvines on small number of variables and can improve the model estimation. Furthermore when some maximal cliques share no separators or a separator with one element only, their corresponding vine structure can be estimated separately. This leads to a great reduction of computational effort when data from a sparse model is considered.

However, it is also very clear that if our assumptions of conditional independence structure present in the data are not correct the performance of our method will not be optimal.

4.2. Heuristics

Algorithm 3 explains how the m-vine corresponding to a given strong clique tree can be constructed. There are still many possible m-vine structures and, in general, it is not known how many there are. In this section, we will design a heuristic search for the 'best' m-vine structure for the data. This will be achieved by specifying information on how choices should be made in Algorithm 3 in every step where a choice is possible. We discuss all these choices below.

- *In line 5 in Algorithm 3, the method of constructing a regular vine on given elements is required.*
  At this point we will use Dißmann's heuristic where a vine structure is constructed tree by tree and each tree structure is determined by the maximum spanning tree with absolute value of Kendall's tau being the edge weight. Hence in the case of only one maximal clique, the m-vine approach coincides with Dißmann's heuristic.
- *In line 8 in Algorithm 3, the way of combining vines corresponding to separators and the extra elements needs to be provided.*
  First we merge the vines for the separators and then extend the merger by the extra elements. The order of merging separators is determined by Algorithm 5. We require two choices at this point.
  1. The choice of the sampling order for the regular vines in Algorithm 2 when vines without overlap are merged and tree structures for trees higher than T_p when merging vines with overlap.
     When vines do not overlap, the sampling order is chosen based on empirical multiple correlation (Yule and Kendall, 1965). For n variables and their empirical correlation matrix Σ of normal scores, the empirical multiple correlation of variable i is $R^2_{i\{\Sigma\}} = 1 - \frac{det(\Sigma)}{det(\Sigma^{ii})}$, where Σ^{ii} denotes the matrix Σ without ith row and ith column and det is the determinant.
     When merging $V(n)$ and $V(m)$ where $\mathcal{V}_n \cap \mathcal{V}_m = \emptyset$, the sampling order $SO_n = \{a_1, \ldots, a_n\}$ is determined in reverse. From the top node of $V(n)$ there are two variables in the conditioned set denoted as v_n, v_{n-1}. The empirical multiple correlation for these two variables are denoted by $R^2_{v_n\{\Sigma\}}$ and $R^2_{v_{n-1}\{\Sigma\}}$ respectively, where Σ is the empirical correlation matrix for variables $\mathcal{V}_n \cup \mathcal{V}_m$. Since a_n will appear in the top node of the merger, it's more reasonable to choose a_n with smaller correlation in higher trees. When the value of a_n is determined, $V(n)$ is marginalized by a_n and we get a subvine with conditioned set $v_{n-1}v_{n-2}$ in its top node. The variable a_n will be removed from the matrix Σ. By successively applying this approach we get a reverse sampling order $(a_n, \ldots, a_1)$. The sampling order of $V(m)$ can be similarly determined. The algorithm below concludes the above procedure.

Algorithm 4 Determine a sampling order of $V(n)$ during merging with $V(m)$.

Input: A regular vine $V(n)$ and empirical correlation matrix $\Sigma(\mathcal{V}_n \cup \mathcal{V}_m)$
Output: A sampling order $SO_n = (a_1, \ldots, a_n)$

1: **for** i from n to 2 **do**
2: Calculate the empirical multiple correlation $R^2_{v_i\{\Sigma\}}$ and $R^2_{v_{i-1}\{\Sigma\}}$ for the variables v_i, v_{i-1} in the conditioned set of the top node of $V(n)$.
3: a_i is chosen to be the variable with smaller empirical multiple correlation.
4: $V(n)$ is updated by marginalizing a_i.
5: Σ is updated by removing the row and the column of a_i.
6: **end for**

When merging $V(n)$ and $V(m)$ with overlapping elements $\mathcal{V}_p \neq \emptyset$, the heuristic to determine the tree structure higher than T_p is the same as Dißmann's heuristic. A weight (absolute value of Kendall's tau) is assigned to the possible edges and the tree structure is chosen to be the maximum spanning tree.

2. When there is only one extra element, then vine structure can be constructed by extending a vine by this element as explained in Algorithm 2. However, when there are more extra elements, we must decide in which order they should be added. We propose to construct a vine on extra elements using Dißmann's heuristic and merge these vines using the approach explained in the previous point.

Note that due to the choices we made, not all m-vine structures are considered. This is mainly due to the choice we made in the second point. For example in Fig. 10, the vine $V(\{1, 4, 5, 9, 10\})$ is constructed by at first fixing a vine structure for $V(\{1, 4, 5\})$ and then merging with $V(\{9, 10\})$, which accounts for only 8 possibilities. This choice eliminates 40 possible vine structures for $V(\{1, 4, 5, 9, 10\})$.

4.3. Simulation

A simulation study is presented in this section to show the performance of the proposed heuristic when compared to the Dißmann's heuristic. The data is simulated from a regular vine copula model. The vine structure is randomly constructed by the function `RVineMatrixSample`. Its bivariate copula families are chosen from Gumbel(G), Clayton(C) and Joe(J) copulas as well as their rotated versions. The copula parameters are obtained from Kendall's tau which are simulated from a *Beta(2,2)* distribution. In order to distinguish from the independence copula, all Kendall's tau smaller than 0.2 are rounded off to 0.2 (according to the test in Genest and Favre (2007) this 0.2 Kendall's tau is rejected to be 0 for 300 or 1000 sample size). The Kendall's tau is also allowed to take negative values with probability 0.5.

The conditional independence are introduced by at first determining a truncation level k and then replacing randomly the bivariate copulas in tree T_k with independence copula. The dimension of the simulated model is 10, 20 and 30 respectively and the sampled data size is 300 or 1000. The proportions of the independent copulas in vine models are chosen to be 30% (leading to a not very sparse model) and 70% (sparse model). Hence, for dimension 10 there are 13 ($\lfloor 45 * 0.3 \rfloor$) or 31 ($\lfloor 45 * 0.7 \rfloor$) independence copulas assigned, for dimension 20 there are 57 (190*0.3) or 133 (190*0.7) independence copulas, and for dimension 30 there are 130 ($\lfloor 435 * 0.3 \rfloor$) or 304 ($\lfloor 435 * 0.7 \rfloor$) independence copulas.

For each simulated data, we apply 1) **method-1**, the Dißmann's heuristic to get a truncated vine; 2) **method-2**, the procedure introduced in the previous section (estimate a graph by `selectFast`, make it strongly chordal if needed) and the heuristic to get an m-vine from the estimated strongly chordal graph; 3) **method-3**, the heuristic directly on the strongly chordal graph from the simulated m-vine model (which is a graph by connecting all the variables in the constraint set of each maximal node of the m-vine respectively). We then calculate the AIC and BIC for the vines estimated by these three approaches. Furthermore, we apply a Vuong test (without or with Schwarz correction) to the estimated vines to see whether one vine is significantly better than the other. This procedure is repeated 100 times. The results are shown in Table 4.

We draw several conclusions from the simulation results.

- The performance of method-2 is comparable to method-1 when the simulated data is not very sparse (30% proportion of independence copula) in dimension 10 and 20. However, when the simulated data is sparse (70% proportion of independence copula) method-2 performs worse than method-1 especially when the dimension is high and the data size is small. This is because the data is simulated from a non-Gaussian distribution which violates the Gaussian assumption used in the function `selectFast`. Moreover, when smaller data size is generated, this function is less likely to capture the conditional independence well especially in high dimensions. According to the Vuong test, the m-vines from method-2 and method-3 are less likely to have similar performance when the underlying data is sparse;
- when the underlying strongly chordal graph is correctly captured in method-3, the performance is much better than method-1 when the simulated data is sparse. However in dimension 20 when the underlying data is not very sparse or in dimension 30, the performance of method-3 is not as good as in other cases. This is due to the choice of our heuristic vine structure constructions, where lots of vine structures are not considered (as discussed in the example in Section 4.2). Even though the vine structures are not well chosen in general, one can still find some vine structures with much better AIC or BIC, as shown in Table 4 where the average improvement in AIC and BIC is large.

5. Real data analysis

In this section, we analyze two data sets.

Data set 1. The first data set we consider is the uranium data set (Cook and Johnson, 1986). It consists of 655 data points for 7 variables, which are denoted as X_1 (Uranium), X_2 (Lithium), X_3 (Cobalt), X_4 (Potassium), X_5 (Cesium), X_6 (Scandum) and X_7 (Titanium). The data is transformed into standard uniform scale using their empirical distributions. The function `selectFast` is applied to the data transformed to standard normal scale and we get that the output graph is not chordal. After two extra edges 16 and 27 are added into the graph, the final strong clique tree is obtained, which is shown in Fig. 13.

We can observe that rather than estimating a dim-7 regular vine one can analyze this data by estimating two dim-5 vines and one dim-4 vine with overlapping elements. Following our heuristic m-vine approach we obtain a simplified m-

Table 4
Simulation result in dimension 10, 20 and 30 respectively with proportion of independence copula being 30% or 70%, with 300 or 1000 data size. M-2 denotes method-2 and M-3 denotes method-3. *#imp* denotes the number of times the estimated m-vine and the truncated vine from method-1 are not significantly different according to the Vuong test or the estimated m-vine is significantly better (the number for this case is shown in brackets). ΔAIC and ΔBIC denotes the average absolute AIC and BIC improvement when the estimated m-vine is significantly better than the truncated vine. $\#imp_{Schwarz}$, $\Delta AIC_{Schwarz}$ and $\Delta BIC_{Schwarz}$ correspond to the same results of the Vuong test with Schwarz correction. $\#OVLP$ denotes the number of times the m-vines from method-2 and method-3 are not significantly different according to the Vuong test (the number in brackets correspond to the Vuong test with Schwarz correction).

Dim	%	size		#imp	ΔAIC	ΔBIC	$\#imp_{Schwarz}$	$\Delta AIC_{Schwarz}$	$\Delta BIC_{Schwarz}$	$\#OVLP$
10	30	300	M-2	71(33)	267.88	262.94	71(25)	345.71	342.30	17(18)
			M-3	80(67)	1129.93	1175.32	84(70)	1088.23	1133.58	
		1000	M-2	81(34)	586.96	585.51	81(32)	622.37	621.91	7(5)
			M-3	84(76)	4067.88	4162.23	86(80)	3877.10	3970.65	
	70	300	M-2	49(30)	352.33	354.67	49(30)	352.33	354.67	7(7)
			M-3	96(84)	448.61	467.48	96(86)	439.47	458.55	
		1000	M-2	56(39)	901.21	908.88	54(39)	904.02	912.58	8(7)
			M-3	92(89)	1545.87	1588.94	96(89)	1545.87	1588.94	
20	30	300	M-2	84(73)	406.15	193.32	74(29)	609.99	380.35	15(16)
			M-3	59(40)	1604.54	1744.54	64(50)	1330.61	1474.39	
		1000	M-2	72(30)	441.84	436.44	77(30)	450.86	450.86	8(11)
			M-3	61(50)	4408.98	4781.87	64(56)	3983.06	4356.40	
	70	300	M-2	31(18)	925.64	883.16	29(18)	926.89	894.39	3(0)
			M-3	99(96)	2943.62	3249.96	100(97)	2915.97	3222.12	
		1000	M-2	81(71)	842.96	641.15	75(37)	1341.46	1170.62	1(0)
			M-3	99(98)	12268.93	12979.95	99(99)	12147.05	12857.28	
30	30	300	M-2	82(73)	599.97	189.19	59(8)	997.39	420.53	9(18)
			M-3	47(25)	2036.46	2226.09	58(34)	1604.84	1813.34	
		1000	M-2	25(13)	747.70	678.93	22(14)	701.45	663.88	16(19)
			M-3	67(45)	442.24	302.50	41(11)	1141.46	897.68	
	70	300	M-2	49(42)	574.41	231.53	34(3)	1039.37	491.21	10(7)
			M-3	64(57)	3158.75	3957.02	82(71)	2592.34	3376.34	
		1000	M-2	94(90)	883.60	390.24	84(32)	1341.17	771.41	2(6)
			M-3	73(72)	12479.21	14267.61	79(73)	12312.03	14101.68	

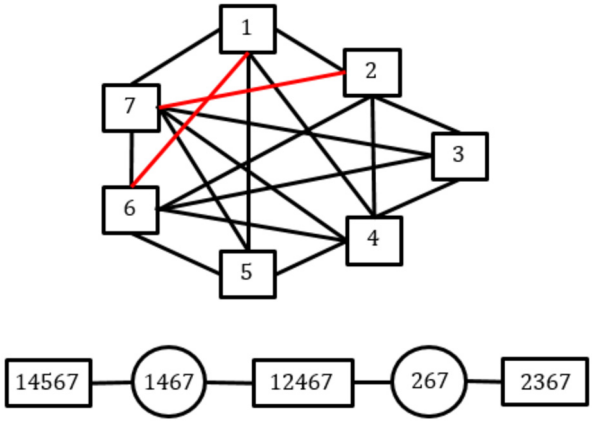

Fig. 13. The strongly chordal graph (by adding the extra edges marked in red) and the strong clique tree for uranium data set.

vine with $AIC = -1655.24$ and $BIC = -1534.16$. The Dißmann's heuristic gives a 3-truncated vine with $AIC = -1572.59$ and $BIC = -1496.35$. The m-vine approach is better (the p-value of the Vuong test is equal to 8.19E-03 but 0.3302 with Schwarz correction) and it can be improved further by taking into account non-simplified vines. This data set was already extensively studied in Acar et al. (2012); Killiches et al. (2017) and there has been shown that the conditional copula $C_{36|7}$ does depend on the conditioning variable.

In our further analysis we concentrate on the maximal clique $\{2, 3, 6, 7\}$ and the vine structures for other maximal cliques can be constructed similarly. The best vine structure for the maximal clique $\{2, 3, 6, 7\}$ can be found similarly as in Section 4.1, which is to search for all possible structures, estimate simplified vine at first and non-simplified vine instead if it's rejected according to the `pacotest`. Node 67 has to be included in the first tree as the restriction in the graph. By extending $V(\{6, 7\})$ with variable 2 and 3 there are 12 possible vine structures. For each of them, we apply the mentioned

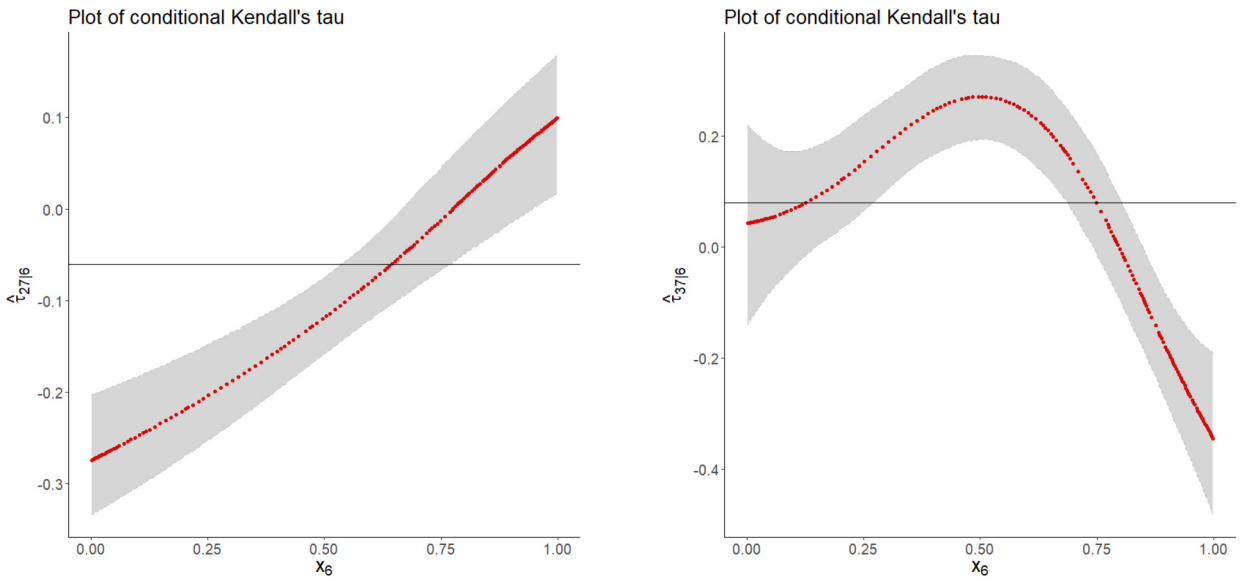

Fig. 14. Kendall's tau for the conditional copula $C_{27|6}$ (left) and $C_{37|6}$ (right), marked in red points with its 95% confidence interval. The horizontal line denotes the conditional Kendall's tau of the simplified copula.

estimation procedure and find the best one according to AIC ($AIC = -939.50$) is a non-simplified vine. This vine has the same vine structure as the simplified subvine on $\{2, 3, 6, 7\}$ in the heuristic m-vine approach (with $AIC = -891.88$ and $BIC = -847.03$). It contains nodes 26, 67 and 36 in the second tree, nodes 27|6 and 37|6 in the third tree and node 23|67 in the last tree. We see that non-simplified vines are better models for the data. The Kendall's tau for the conditional copula in the third tree for both simplified and non-simplified copula are shown in Fig. 14. The best non-simplified vine according to BIC ($BIC = -863.40$) is the one with nodes 26, 67, 23 in the second tree, 27|6, 36|2 in the third tree and 37|26 in the fourth tree.

Data set 2. The second data set is the financial data in *Kenneth R.French - Data library* (http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html) which consists of 1142 monthly returns of 49 industry portfolios from 07-1926 to 08-2021. Detailed explanation for the industries can be found in Appendix A.6. We follow the steps taken in Kurz (2019) and first filter the data by a ARMA$(1, 1)$-GARCH$(1, 1)$ model with Student's t innovation. The residuals are fitted by their empirical distributions, and the fitted margins are then transformed to standard uniforms. The graph for the data in standard normal scale given by the function `selectFast` consists of 220 edges and is not chordal. To make this graph chordal extra 358 edges are added, and furthermore in order to make it strongly chordal 101 more edges are added. The final strongly chordal graph is shown in Fig. 15 by the function `tkplot` (Csardi and Nepusz, 2006) and the added extra edges are marked in red. The obtained twenty maximal cliques in the corresponding clique tree are shown in Appendix A.6.

Although there were many edges added to make the graph strongly choral, for some indices their relationship to other indices stays the same, for example the index **Mines** for which no extra edges are added to have it connected with other vertices. Furthermore not all the added edges are meaningless to represent the relationship between indices. For example, the index **Oil** was connected with indices **Chems**, **Cnstr**, **Mach**, **Mines**, **Coal**, **Util** and **Fin**, and thirteen more edges between **Oil** and other indices are added like with **Fun** and **Steel**. In our m-vine approach, the relationship between **Oil** and **Fun**, and between **Oil** and **Steel** are estimated by a G180 copula ($\tau = 0.33$, lower tail dependence $\lambda_L = 0.41$) and a BB1 copula ($\tau = 0.44$, lower tail dependence $\lambda_L = 0.48$ and upper tail dependence $\lambda_U = 0.30$) respectively. Discussion about how the **Oil** industry affects the **Fun** industry can be found in Dahlquist and Vonderau (2022). The relationship between **Oil** industry and **Steel** industry seems more intuitive. There is a stronger lower tail dependence than the upper tail dependence for these variables.

The resulting m-vine has $AIC = -52749.33$ and $BIC = -49362.09$. Compared with the estimated 28-truncated vine from Dißmann's heuristic ($AIC = -52217.30$ and $BIC = -48930.87$), the m-vine approach is significantly better (the p-value in Vuong test equals to 2.10E-4, and 5.20E-3 with Schwarz correction). In the constructed strong clique tree, there are some maximal cliques with five or six variables for which we can further improve the estimation by assessing all vine structures corresponding to those maximal cliques.

6. Conclusion

In this paper we studied m-vines introduced in Kurowicka and Cooke (2006) and proved their relationship to the strongly chordal graphs. We proposed to use this result to simplify the regular vine copula model by assuming that a pattern of

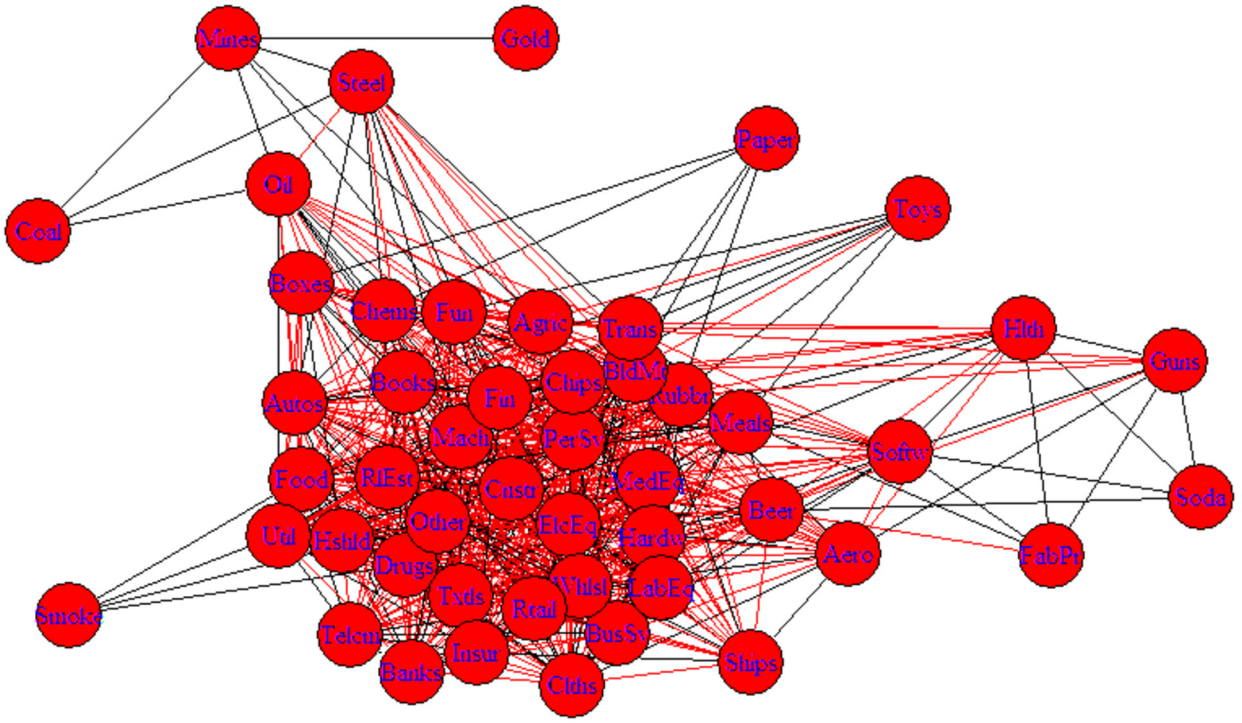

Fig. 15. The final constructed strongly chordal graph. The extra edges added to make the graph strongly chordal is marked as red.

conditional independence represented by strongly chordal graph is appropriate for the data. This makes the estimation of the model much less expensive. When modeling a sparse data it allows to consider all possible vine structures corresponding to nodes in strong clique tree, and furthermore it makes feasible to consider non-simplified vines.

We showed that when the assumptions of the pattern of conditional independence is correct our method provides obvious advantages. However, it is not surprising that when the model is simplified incorrectly, our method might miss important dependencies and its performance might be poor.

We summarize several deficiencies of the m-vine approach as follows:

- The estimation accuracy of the m-vine approach relies heavily on the initial graph generated from the data as shown in the simulation study in Section 4.3. The m-vine approach will not consider more complex vine structures than the information about conditional (in)dependencies given by the graph. In this paper, we use the Gaussian graphical model to generate the graph. When non-Gaussian dependence appears, methods in Bauer et al. (2012); Bauer and Czado (2016); Hobæk Haff et al. (2016) (where the conditional independence is captured by pair copula constructions including vines) can be used. This issue is however still an open problem.
- The m-vine approach benefits from the sparsity of given data to reduce computations. However, in some cases many edges may be added into the estimated graph in order to make it strongly chordal, for example when the estimated graph is sparse but contains a large cycle. This problem so far cannot be mitigated.
- Even though we can assess all possible structures for the subvines, the true m-vine structure may not be found. This is due to the restrictions in the strongly chordal graph or the heuristics we proposed. However, we can assess all possible subvine structures and choose the best if maximal cliques with small number of variables are obtained.
- To the best of the authors' knowledge, it is not known in general how specified conditional independence can be represented by a regular vine with some nodes removed. In this paper we only discussed the conditional independence represented by strongly chordal graphs.

Acknowledgement

Kailun Zhu is supported by the China Scholarship Council. We would like to thank the associate editor and three anonymous referees for their valuable suggestions. We would also like to thank Gabriela F. Nane for her contribution to improving the text in the paper.

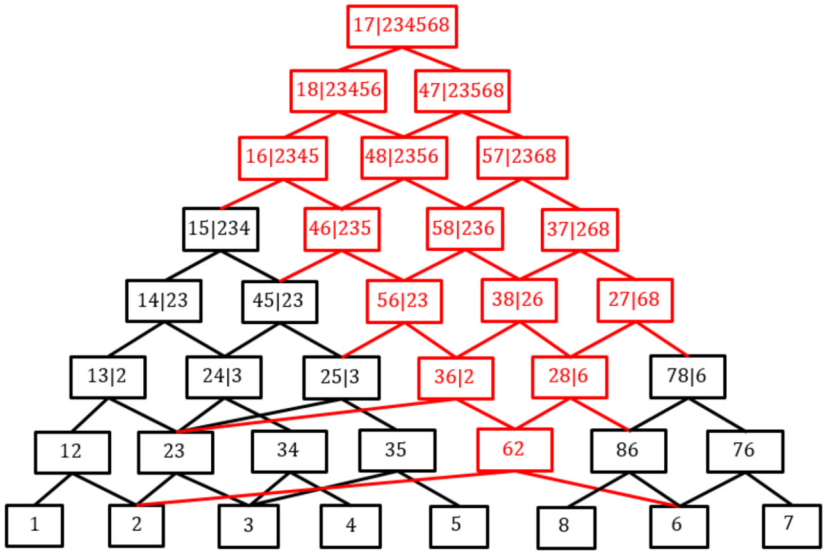

Fig. 16. A merger of vine $V_1(5)$ ($V(\{1,2,3,4,5\})$) (black) and $V(3)$ ($V(\{6,7,8\})$ (black). The new nodes and lines in the vine triangular array are marked in red.

Appendix A

A.1.

Below we show an example of how to merge two regular vines without overlap. One vine is $V_1(5)$ ($V(\{1,2,3,4,5\})$) and the other vine is $V(3)$ ($V(\{6,7,8\})$), both of them shown in black in Fig. 16. The sampling order we have chosen for $V(5)$ is $(2,3,5,4,1)$ and for $V(3)$ it is $(6,8,7)$.

We see that the partners of $a_5 = 1$ are 7 in T_8, 8 in T_7 and 6 in T_6. Then marginalizing by 1 we have the partners of $a_4 = 4$ are 7 in T_7, 8 in T_6 and 6 in T_5. Through marginalizing by $a_3 = 5$, $a_2 = 3$ and $a_1 = 2$, we obtain their partners and the structure of joint vine with new nodes printed in red is shown in Fig. 16.

A.2.

The theorem in this section concerns constraints on sampling orders that will lead to consistent mergers of two vines with overlap using the method in Algorithm 2.

Theorem A.1. *Given two regular vines $V(n)$ and $V(m)$ with $\mathcal{V}_n \cap \mathcal{V}_m = \mathcal{V}_p \neq \emptyset$, the merger $V(\mathcal{V}_n \cup \mathcal{V}_m)$ constructed by two sampling orders $SO_n = (a_1, \ldots, a_n)$ and $SO_m = (b_1, \ldots, b_m)$ according to Algorithm 2 is valid (leads to a regular vine on elements $\mathcal{V}_n \cup \mathcal{V}_m$) if and only if $(a_1, \ldots, a_p)$ and $(b_1, \ldots, b_p)$ are sampling orders of $V(p)$.*

Proof. The structure of the merger is fixed up to and including tree T_p hence only the segment $(a_{p+1}, \ldots, a_n)$ in SO_n and $(b_{p+1}, \ldots, b_m)$ in SO_m affect the construction of the merger. According to Algorithm 2, in the merger, the new partners of a_i, $p+1 \leq i \leq n$ are $(b_m, \ldots, b_{p+1})$ from tree T_{i+m-p} to T_{i+1}, and the new partners of b_j, $p+1 \leq j \leq m$ are $(a_n, \ldots, a_{p+1})$ from tree T_{j+n-p} to T_{j+1}. Denote the top node of $V(p)$ as t_p.

Sufficiency: The proof of sufficiency is by construction according to Algorithm 2. Because $(a_1, \ldots, a_p)$ and $(b_1, \ldots, b_p)$ are sampling orders of the common subvine $V(p)$, the segment $(a_{p+1}, \ldots, a_n)$ and $(b_{p+1}, \ldots, b_m)$ do not contain elements in $\mathcal{V}_p$. Hence elements $(a_{p+1}, \ldots, a_n)$ are not in $\mathcal{V}_m$ and $(b_{p+1}, \ldots, b_m)$ are not in $\mathcal{V}_n$. This means that construction of the merger up to and including tree T_{p+2} is valid as it follows a vine structure construction for merger of two vines without overlap. A new node with conditioned set $a_{p+1}b_{p+1}$ is chosen in T_{p+2} of the merger, which has one child in $V(n)$ (whose conditioned set contains a_{p+1}) and the other child in $V(m)$ (whose conditioned set contains b_{p+1}). Since $(a_1, \ldots, a_p)$ and $(b_1, \ldots, b_p)$ are sampling orders of $V(p)$, these two children are both a parent node of t_p. Hence the tree construction in T_{p+1} is also valid, and a consistent merger is obtained.

Necessity: Let us assume that only $(a_1, \ldots, a_p)$ is not a sampling order of $V(p)$. Hence at least one element in the conditioned set $\{s,t\} \in \mathcal{V}_p$ of t_p, is in $(a_{p+1}, \ldots, a_n)$. Let $a_k = s$ (the case that $a_k = t$ is similar), $p+1 \leq k \leq n$, and the new partners of a_k in the merger are $(b_m, \ldots, b_{p+1})$ from tree T_{k+m-p} to tree T_{k+1}. We also have that a_k is either in the conditioning or conditioned set of parents of t_p in tree T_{p+1} of $V(m)$. If a_k is in the conditioning set, then by proximity condition it cannot be in the conditioned sets of nodes in higher trees. If a_k is in the conditioned set, there must be a parent node where a_k

is partnered with b_{p+1}, hence node with conditioned set $a_k b_{p+1}$ already exists in $V(m)$. This leads to contradiction as each pair of elements exists in the conditioned set of a node only once in a regular vine. □

A.3.

See Fig. 17.

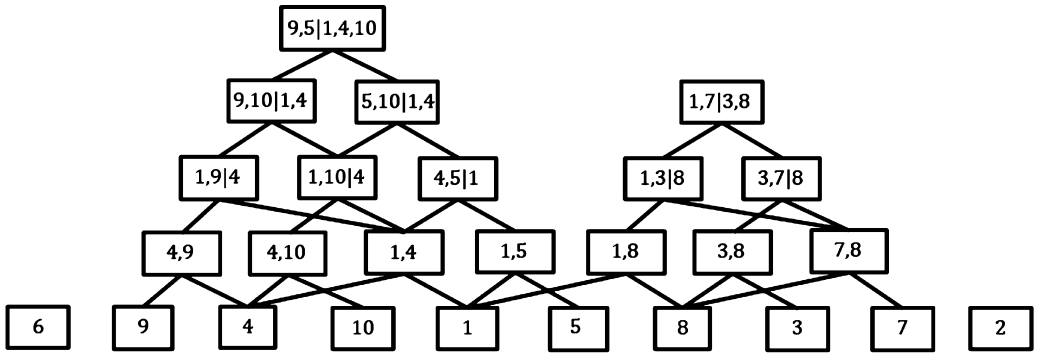

Fig. 17. The vine triangular array representation for the m-vine in Section 4.1.

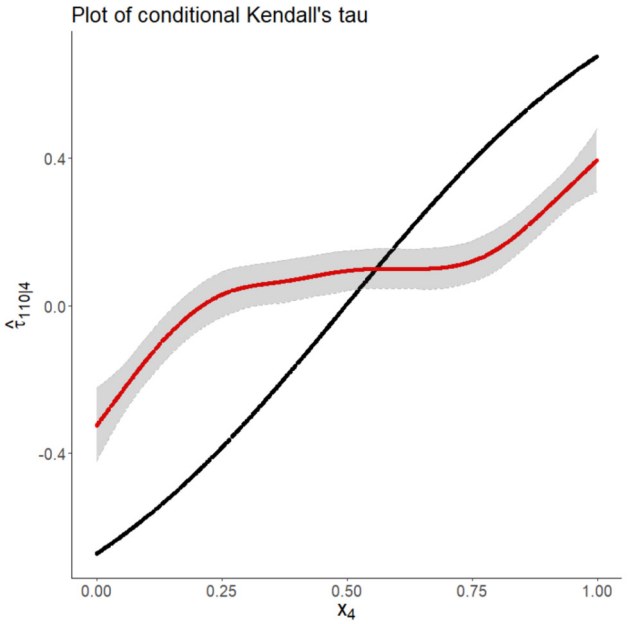

Fig. 18. Kendall's tau for the conditional copula $C_{1,10|4}$ with respect to X_4. The true conditional tau is marked in black points and the estimated tau is marked in red points with its 95% confidence interval.

A.4.

To show the dependence between X_1 and X_{10} conditional on X_4, we simulated data from the estimated m-vine. The relationship between the conditional Kendall's tau of $C_{1,10|4}$ (copula family is C180 or C270) and variable X_4, estimated by GAM model is shown in the figure below. The estimated non-simplified m-vine where this relationship is not modeled directly cannot recover the relationship quite well. See Fig. 18.

A.5.

In line 13 in Algorithm 3 when all the vines corresponding to the nodes $\mathcal{T}^{(0)}$ are combined, the order of combining these vines should follow Algorithm 5 as well. This is achieved by setting $\mathcal{S}$ to be the maximal cliques $\mathcal{C}$ of the strongly chordal graph and the underlying tree structure is $\mathcal{T}^{(0)}$.

Algorithm 5 Determine the order of separators to be merged for node $C_j^{(i)}$.

Input: the separators $\mathcal{S}$ and the clique tree $\mathcal{T}^{(i+1)}$
Output: the order of separators *order*

$index = 0$.
while not all separators in $\mathcal{S}$ are considered **do**
  Randomly pick one separator from the separators that haven't been considered, denoted as $\mathcal{S}'_{index}$.
  Set $index = index + 1$ and $order[index] =$ the index of $\mathcal{S}'_{index}$ in $\mathcal{S}$.
  Find the neighbors of $\mathcal{S}'_{index}$ in $\mathcal{T}^{(i+1)}$, denoted as $neigh_{index}$.
  Apply the below recursive procedure.
  procedure RECURSE($neigh_{index}$)
    if there are at least one neighbor in $neigh_{index}$ **then**
      for each neighbor in $neigh_{index}$ **do**
        if this neighbor is in $\mathcal{S}$ **then**
          if this neighbor has overlap with $\mathcal{S}'_{index}$ **then**
            Denote this neighbor as $\mathcal{S}''_{index}$.
            Set $index = index + 1$, $order[index] =$ the index of $\mathcal{S}''_{index}$ in $\mathcal{S}$.
          end if
        end if
        Find the neighbors of this neighbor in $\mathcal{T}^{(i+1)}$ and choose ones which haven't been considered before, denoted as $neigh_{index}$.
        RECURSE($neigh_{index}$)
      end for
    end if
  end procedure
end while

A.6.

The industries in the 49 industry portfolios are, Agriculture (**Agric**, X_1), Food Products (**Food**, X_2), Candy & Soda (**Soda**, X_3), Beer &Liquor (**Beer**, X_4), Tobacco Products (**Smoke**, X_5), Recreation (**Toys**, X_6), Entertainment (**Fun**, X_7), Printing &Publishing (**Books**, X_8), Consumer Goods (**Hshld**, X_9), Apparel (**Clths**, X_{10}), Healthcare (**Hlth**, X_{11}), Medical Equipment (**MedEq**, X_{12}), Pharmaceutical Products (**Drugs**, X_{13}), Chemicals (**Chems**, X_{14}), Rubber and Plastic Products (**Rubbr**, X_{15}), Textiles (**Txtls**, X_{16}), Construction Materials (**BldMt**, X_{17}), Construction (**Cnstr**, X_{18}), Steel Works Etc (**Steel**, X_{19}), Fabricated Products &Machinery (**FabPr**, X_{20}), Machinery (**Mach**, X_{21}), Electrical Equipment (**ElcEq**, X_{22}), Automobiles &Trucks (**Autos**, X_{23}), Aircraft (**Aero**, X_{24}), Ships &Railroad Equipment (**Ships**, X_{25}), Defense (**Guns**, X_{26}), Precious Metals (**Gold**, X_{27}), Non-Metallic &Industrial Metal Mining (**Mines**, X_{28}), Coal (**Coal**, X_{29}), Petroleum &Natural Gas (**Oil**, X_{30}), Utilities (**Util**, X_{31}), Communication (**Telcm**, X_{32}), Personal Services (**PerSv**, X_{33}), Business Services (**BusSv**, X_{34}), Computers (**Hardw**, X_{35}), Computer Software (**Softw**, X_{36}), Electrical Equipment (**Chips**, X_{37}), Measuring & Control Equipment (**LabEq**, X_{38}), Business Supplies (**Paper**, X_{39}), Shipping Containers (**Boxes**, X_{40}), Transportation (**Trans**, X_{41}), Wholesale (**Whlsl**, X_{42}), Retail (**Rtail**, X_{43}), Restaurants, Hotels &Motels (**Meals**, X_{44}), Banking (**Banks**, X_{45}), Insurance (**Insur**, X_{46}), Real Estate (**RlEst**, X_{47}), Trading (**Fin**, X_{48}) and Almost nothing (**Other**, X_{49}).

The maximal cliques in the strong clique tree for the financial data set in Section 5.
{27, 28}
{19, 28, 29, 30}
{3, 4, 11, 26, 36}
{1, 7, 19, 28, 30}
{2, 5, 13, 31, 49}
{14, 15, 17, 39, 40, 41}
{4, 11, 15, 20, 26, 36}
{4, 11, 15, 17, 24, 26, 36, 41}
{1, 6, 7, 8, 15, 17, 37, 41, 44}
{1, 4, 11, 12, 15, 17, 24, 33, 36, 37, 41}
{1, 7, 8, 14, 15, 17, 19, 21, 23, 30, 33, 37, 41, 48}
{1, 2, 7, 8, 13, 14, 15, 17, 18, 21, 23, 30, 31, 33, 37, 41, 48, 49}
{1, 4, 12, 13, 15, 17, 18, 21, 22, 24, 33, 34, 35, 36, 37, 38, 41, 43, 48, 49}
{1, 4, 10, 12, 13, 15, 17, 18, 21, 22, 24, 25, 33, 34, 35, 37, 38, 41, 42, 43, 48, 49}
{1, 2, 7, 8, 9, 12, 13, 14, 15, 17, 18, 21, 22, 23, 31, 33, 37, 40, 41, 43, 48, 49}
{1, 4, 10, 12, 13, 15, 17, 18, 21, 22, 25, 33, 34, 35, 37, 38, 41, 42, 43, 46, 47, 48, 49}
{1, 2, 7, 8, 9, 12, 13, 14, 15, 17, 18, 21, 22, 23, 31, 33, 34, 35, 37, 38, 41, 42, 43, 46, 47, 48, 49}
{1, 2, 7, 8, 9, 12, 13, 15, 16, 17, 18, 21, 22, 23, 31, 33, 34, 35, 37, 38, 41, 42, 43, 45, 46, 47, 48, 49}
{1, 2, 4, 7, 8, 9, 10, 12, 13, 15, 16, 17, 18, 21, 22, 33, 34, 35, 37, 38, 41, 42, 43, 44, 46, 47, 48, 49}
{1, 2, 4, 7, 8, 9, 10, 12, 13, 15, 16, 17, 18, 21, 22, 31, 32, 33, 34, 35, 37, 38, 41, 42, 43, 45, 46, 47, 48, 49}

Appendix B. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.csda.2022.107461.

References

Aas, K., Czado, C., Frigessi, A., Bakken, H., 2009. Pair-copula constructions of multiple dependence. Insur. Math. Econ. 44, 182–198. https://doi.org/10.1016/j.insmatheco.2007.02.001.

Acar, E.F., Craiu, R.V., Yao, F., 2011. Dependence calibration in conditional copulas: a nonparametric approach. Biometrics 67, 445–453.

Acar, E.F., Czado, C., Lysy, M., 2019. Flexible dynamic vine copula models for multivariate time series data. Econom. Stat. 12, 181–197. https://doi.org/10.1016/j.ecosta.2019.03.002. https://www.sciencedirect.com/science/article/pii/S2452306219300206.

Acar, E.F., Genest, C., Nešlehová, J., 2012. Beyond simplified pair-copula constructions. J. Multivar. Anal. 110, 74–90. https://doi.org/10.1016/j.jmva.2012.02.001.

Bauer, A., Czado, C., 2016. Pair-copula bayesian networks. J. Comput. Graph. Stat. 25, 1248–1271. https://doi.org/10.1080/10618600.2015.1086355.

Bauer, A., Czado, C., Klein, T., 2012. Pair-copula constructions for non-gaussian dag models. Can. J. Stat. 40, 86–109. https://doi.org/10.1002/cjs.10131.

Bedford, T., Cooke, R., 2001. Probability density decomposition for conditionally dependent random variables modeled by vines. Ann. Math. Artif. Intell. 32, 245–268. https://doi.org/10.1023/A:1016725902970.

Bedford, T., Cooke, R., 2002. Vines - a new graphical model for dependent random variables. Ann. Stat. 30, 1031–1068. https://doi.org/10.1214/aos/1031689016.

Brechmann, E.C., Czado, C., Aas, K., 2012. Truncated regular vines in high dimensions with application to financial data. Can. J. Stat. 40, 68–85. http://www.jstor.org/stable/41724516.

Cook, R.D., Johnson, M.E., 1986. Generalized Burr-Pareto-logistic distributions with applications to a uranium exploration data set. Technometrics 28, 123–131. https://doi.org/10.2307/1270448.

Cooke, R., Kurowicka, D., Wilson, K., 2015. Sampling, conditionalizing, counting, merging, searching regular vines. J. Multivar. Anal. 138, 4–18. https://doi.org/10.1016/j.jmva.2015.02.001.

Csardi, G., Nepusz, T., 2006. The igraph software package for complex network research. Int. J Complex Syst. 1695. https://igraph.org.

Czado, C., 2019. Analyzing Dependent Data with Vine Copulas. Lecture Notes in Statistics. Springer.

Dahlquist, M., Vonderau, P., 2022. Petrocinema: Sponsored Film and the Oil Industry. Bloomsbury Academic. https://books.google.nl/books?id=bUBBEAAAQBAJ.

Dißmann, J., Brechmann, E., Czado, C., Kurowicka, D., 2013. Selecting and estimating regular vine copulae and application to financial returns. Comput. Stat. Data Anal. 59, 52–69. https://doi.org/10.1016/j.csda.2012.08.010.

Farber, M., 1983. Characterizations of strongly chordal graphs. Discrete Math. 43, 173–189. https://doi.org/10.1016/0012-365X(83)90154-1.

Friedman, J., Hastie, T., Tibshirani, R., 2007. Sparse inverse covariance estimation with the graphical lasso. Biostatistics 9, 432–441. https://doi.org/10.1093/biostatistics/kxm045.

Gavril, F., 1974. The intersection graphs of subtrees in trees are exactly the chordal graphs. J. Comb. Theory, Ser. B 16, 47–56. https://doi.org/10.1016/0095-8956(74)90094-X. https://www.sciencedirect.com/science/article/pii/009589567490094X.

Genest, C., Favre, A.C., 2007. Everything you always wanted to know about copula modeling but were afraid to ask. J. Hydrol. Eng. 12, 347–368. https://doi.org/10.1061/(ASCE)1084-0699(2007)12:4(347).

Giraud, Christophe, Huet, Sylvie, Verzelem, Nicolas, 2009. Graph selection with GGMselect. arXiv:0907.0619.

Hobæk Haff, I., Aas, K., Frigessi, A., Lacal, V., 2016. Structure learning in bayesian networks using regular vines. Comput. Stat. Data Anal. 101, 186–208. https://doi.org/10.1016/j.csda.2016.03.003. https://www.sciencedirect.com/science/article/pii/S0167947316300457.

Killiches, M., Kraus, D., Czado, C., 2017. Examination and visualisation of the simplifying assumption for vine copulas in three dimensions. Aust. N. Z. J. Stat. 59, 95–117. https://doi.org/10.1111/anzs.12182. https://onlinelibrary.wiley.com/doi/abs/10.1111/anzs.12182.

Kurowicka, D., Cooke, R., 2006. Completion problem with partial correlation vines. Linear Algebra Appl. 418, 188–200. https://doi.org/10.1016/j.laa.2006.01.031. https://www.sciencedirect.com/science/article/pii/S0024379506000619.

Kurowicka, D., Joe, H., 2011. Dependence Modeling: Vine Copula Handbook. World Scientific.

Kurz, M.S., 2019. pacotest: testing for partial copulas and the simplifying assumption in vine copulas. R package version 0.3.1.

Kurz, M.S., Spanhel, F., 2017. Testing the simplifying assumption in high-dimensional vine copulas. e-prints. arXiv:1706.02338.

Lauritzen, S., 1996. Graphical Models. Oxford Statistical Science Series. Clarendon Press.

McKee, T.A., 2003. Subgraph trees in graph theory. Discrete Math. 270, 3–12. https://doi.org/10.1016/S0012-365X(03)00161-4.

Meinshausen, N., Bühlmann, P., 2006. High-dimensional graphs and variable selection with the Lasso. Ann. Stat. 34, 1436–1462. https://doi.org/10.1214/009053606000000281.

Mukhopadhyay, A., Rahman, M.Z., 2021. Algorithms for generating strongly chordal graphs. In: Gavrilova, M.L., Tan, C.K. (Eds.), Transactions on Computational Science XXXVIII. Springer, Berlin, Heidelberg.

Müller, D., Czado, C., 2018. Representing sparse gaussian dags as sparse r-vines allowing for non-gaussian dependence. J. Comput. Graph. Stat. 27, 334–344. https://doi.org/10.1080/10618600.2017.1366911.

Müller, D., Czado, C., 2019a. Dependence modelling in ultra high dimensions with vine copulas and the graphical lasso. Comput. Stat. Data Anal. 137, 211–232. https://doi.org/10.1016/j.csda.2019.02.007. https://www.sciencedirect.com/science/article/pii/S0167947319300568.

Müller, D., Czado, C., 2019b. Selection of sparse vine copulas in high dimensions with the lasso. Stat. Comput. 29, 269–287. https://doi.org/10.1007/s11222-018-9807-5.

Nagler, T., Schepsmeier, U., Stoeber, J., Brechmann, E.C., Graeler, B., Erhardt, T., 2019. VineCopula: statistical inference of vine copulas. https://CRAN.R-project.org/package=VineCopula. R package version 2.2.0.

Nagler, T., Vatter, T., 2020. gamCopula: generalized additive models for bivariate conditional dependence structures and vine copulas. https://CRAN.R-project.org/package=gamCopula. R package version 0.0-7.

Nápoles, O., 2010. Bayesian Belief Nets and Vines in Aviation Safety and Other Applications.

Pearl, J., 1988. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Series in Representation and Reasoning. Elsevier Science. https://books.google.nl/books?id=AvNID7LyMusC.

Sklar, A., 1959. Fonctions de répartition à n dimensions et leurs marges. Publ. Inst. Stat. Univ. Paris 8, 229–231.

Spanhel, F., Kurz, M.S., 2015. The partial vine copula: a dependence measure and approximation based on the simplifying assumption. e-prints. arXiv:1510.06971.

Stöber, J., Joe, H., Czado, C., 2013. Simplified pair copula constructions—limitations and extensions. J. Multivar. Anal. 119, 101–118. https://doi.org/10.1016/j.jmva.2013.04.014.

Vuong, Q.H., 1989. Likelihood ratio tests for model selection and non-nested hypotheses. Econometrica, 307–333. https://doi.org/10.2307/1912557.
Wille, A., Bühlmann, P., 2006. Low-order conditional independence graphs for inferring genetic networks. Stat. Appl. Genet. Mol. Biol. 5. https://doi.org/10.2202/1544-6115.1170.
Yule, G., Kendall, M., 1965. An Introduction to the Theory of Statistics. C. Griffin. https://books.google.nl/books?id=CM1WAAAAYAAJ.