Research Article **Open Access**

Special Issue: Salzburg Workshop on Dependence Models & Copulas

J. Górecki*, M. Hofert, and M. Holeňa

Kendall's tau and agglomerative clustering for structure determination of hierarchical Archimedean copulas

DOI 10.1515/demo-2017-0005
Received December 27, 2016; accepted February 24, 2017

Abstract: Several successful approaches to structure determination of hierarchical Archimedean copulas (HACs) proposed in the literature rely on agglomerative clustering and Kendall's correlation coefficient. However, there has not been presented any theoretical proof justifying such approaches. This work fills this gap and introduces a theorem showing that, given the matrix of the pairwise Kendall correlation coefficients corresponding to a HAC, its structure can be recovered by an agglomerative clustering technique.

Keywords: structure determination, agglomerative clustering, Kendall's tau, Archimedean copula

MSC: 62H99

1 Introduction

Hierarchical (also called nested) Archimedean copulas (HACs) bring to the popular class of Archimedean copulas (ACs) the possibility to have different multivariate margins. This possibility is enabled by their construction through *nesting* of several ACs into each other; see [8, pp. 87]. Such a construction, on the one hand, allows for constructing copula models outperforming other recently popular multivariate copula models, e.g., see [15], where HACs are compared to *pair* and *factor* copulas in an empirical study from risk management. On the other hand, such a construction involves an extra effort when estimating HACs, since the *structure* of a HAC, which represents the way the ACs in this HAC are nested into each other, has to be estimated as another parameter.

We can observe intensive research in this direction, e.g, see [2, 5, 10, 14, 17, 18, 20], which provide answers to the question of how to estimate the structure of a HAC based on a data sample. These answers involve different approaches to the problem. In [17], the authors make use of a possibility to decompose the structure of a HAC to a set of trivariate structures. In [10], the problem is treated as an amended shortest path problem. In [18], an approach based on *supertrees* is presented. The rest of the mentioned articles provides approaches that share the feature that both an agglomerative clustering technique [19] and the Kendall correlation coefficient (Kendall's tau) are involved in the estimation process. In the following, we thus refer to these approaches as to *tau-clustering-based*. E.g., in [14], the pair of random variables corresponding to the highest value of Kendall's tau is clustered and the new cluster is represented by its *Kendall transformation*. In [1], the authors suggest to use the *diagonal transformation* instead of the Kendall's one. In [10], Kendall's

***Corresponding Author: J. Górecki:** Department of Informatics, SBA in Karviná, Silesian University in Opava, Univerzitní náměstí 1934/3, 733 40 Karviná, Czech Republic, E-mail: gorecki@opf.slu.cz
M. Hofert: Department of Statistics and Actuarial Science, University of Waterloo, 200 University Avenue West, Waterloo, ON, Canada, E-mail: marius.hofert@uwaterloo.ca
M. Holeňa: Institute of Computer Science, Academy of Sciences of the Czech Republic, Pod vodárenskou věží 271/2, 182 07 Praha, Czech Republic, E-mail: martin@cs.cas.cz

tau used in the previous two approaches is replaced by a more general *τ-Euclidean* metric. Finally in [2, 5], the authors use an agglomerative clustering technique in which the similarity between two random variables is directly the corresponding Kendall's tau and consider the single-, complete- and average-linkage type of clustering.

A simulation study concerning the efficiency of some of the previously mentioned approaches is provided in [18]. There, the approach denoted `kt_kagg`, originally proposed in [2] and merged with an approach to *collapsing* of HAC structures from [14], has shown the best results in the ability to estimate the true structure of a HAC among the 11 estimators considered. Also note that in [3], the `kt_kagg` approach was improved by a new approach to collapsing HAC structures and *re-estimation* of the parameters, which is confirmed in the simulation study reported therein. In the same article, the authors also show that a structure obtained by this improved `kt_kagg` satisfies the *sufficient nesting condition*, which guarantees that a proper copula results, under relatively weak conditions provided the families of the Archimedean generators in the structure are all the same.

However, despite these desirable properties, there has not been presented any theoretical proof showing that if the matrix of Kendall pairwise correlation coefficients (or another measure of concordance) corresponding to a HAC is known, then such an approach recovers the true structure of this HAC. This work attempts to fill this gap and introduces a theorem showing that if such a matrix is known, the structure determination used in the `kt_kagg` approach leads to the true structure.

This work is structured as follows. Section 2 recalls some basics for ACs and HACs. Section 3 recalls the approach to structure determination mentioned in the previous paragraph, including an algorithm in a pseudocode. In Section 4, the announced theorem is provided. Discussion for the theorem is presented in Section 5 and Section 6 concludes.

2 Hierarchical Archimedean Copulas

Let $\mathbb{I} = [0, 1]$ and $\Psi = \{\psi : [0, \infty] \to \mathbb{I} \mid \psi(0) = 1,\ \psi(\infty) = \lim_{t\to\infty} \psi(t) = 0,\ \psi$ is continuous, non-increasing and strictly decreasing in $[0, \inf\{t \in [0, \infty) \mid \psi(t) = 0\}]\ \}$. Also, let a real function y be called *d-monotone* in (a, b), where $a, b \in \mathbb{R}$ and $d \geq 2$, if it is differentiable up to order $d - 2$, the derivatives satisfy

$$(-1)^k y^{(k)}(x) \geq 0,\ \ k = 0, 1, \ldots, d-2, \tag{1}$$

for any $x \in (a, b)$, and $(-1)^{d-2}y^{(d-2)}$ is non-increasing and convex in (a, b). For $d = 1$, let y be called 1-monotone in (a, b) if it is non-negative and non-increasing there. Note that if a function is d-monotone for all $d \in \mathbb{N}$, then it is called *completely monotone* [9, 13]. As follows from [12], given an *Archimedean generator* $\psi \in \Psi$, the function $C : \mathbb{I}^d \to \mathbb{I}$ such that

$$C(u_1, \ldots, u_d) = \psi(\psi^{-1}(u_1) + \ldots + \psi^{-1}(u_d)) \tag{2}$$

is a d-dimensional *Archimedean copula* (d-AC, also denoted by C_ψ) if, and only if, ψ is d-monotone, where ψ^{-1} is the general inverse of ψ given by $\psi^{-1}(s) = \inf\{t \in [0, \infty] \mid \psi(t) = s\}$, $s \in \mathbb{I}$.

To construct a *hierarchical Archimedean copula* (HAC), one just need to replace some arguments in an AC by other HACs [6, 8]. The formal definition of HACs we will recall in this work is taken from [3], which is mainly motivated by the fact that it explicitly refers to the tree HAC structure through concepts from graph theory and thus no auxiliary tools to formalize HAC structures are needed. Note that in the following, $\#s$ denotes the number of elements of a set s.

Definition 1. Let $d, f \in \mathbb{N}$, $d \geq 2$, $f \in \{1, \ldots, d-1\}$, $(\mathcal{V}, \mathcal{E})$ be a labeled tree with nodes $\mathcal{V} = \{1, \ldots, d+f\}$, edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ and rooted in the node $d + f$. Let the nodes $\{1, \ldots, d\}$ be the *leaves* of $(\mathcal{V}, \mathcal{E})$ and the nodes $\{d + 1, \ldots, d + f\}$, which will be called *forks*, have at least two children each. In connection with $(\mathcal{V}, \mathcal{E})$, the following notation will be used:

- For $v \in \mathcal{V}$, denote by $\wedge(v)$ the set of children of v; thus the cardinality of $\wedge(v)$ fulfills $\# \wedge (v) \geq 2$ for $v \in \{d + 1, \ldots, d + f\}$ (i.e., v being a fork) and $\# \wedge (v) = 0$ for $v \in \{1, \ldots, d\}$ (i.e., for v being a leaf).

- For $S \subset \mathcal{V}$ and $\boldsymbol{x^u} \in \mathbb{I}^{d+f}$, the simplified notation

$$\boldsymbol{x}_S = (x_{v_1}, \ldots, x_{v_{\#S}}), \text{ where } S = \{v_1, \ldots, v_{\#S}\}, \tag{3}$$

will be used with a further simplification $\boldsymbol{x}_v = \boldsymbol{x}_{\{v\}}$ for $v \in \mathcal{V}$.

Finally, let $\lambda : \{d+1, \ldots, d+f\} \to \Psi$ be a labeling of forks with Archimedean generators such that for each $\boldsymbol{u} \in \mathbb{I}^d$, there exists $\boldsymbol{x^u} \in \mathbb{I}^{d+f}$ with the following two properties:

(i) $\forall v \in \{d+1, \ldots, d+f\},\ \boldsymbol{x}^{\boldsymbol{u}}_v = C_{\lambda(v)}(\boldsymbol{x}^{\boldsymbol{u}}_{\wedge(v)})$;
(ii) $\forall v \in \{1, \ldots, d\},\ \boldsymbol{x}^{\boldsymbol{u}}_v = u_v$.

Then:

a) if the function $C_{(\mathcal{V},\mathcal{E},\lambda)} : \mathbb{I}^d \to \mathbb{I}$, defined

$$\forall \boldsymbol{u} \in \mathbb{I}^d,\ C_{(\mathcal{V},\mathcal{E},\lambda)}(\boldsymbol{u}) = \boldsymbol{x}^{\boldsymbol{u}}_{d+f}, \tag{4}$$

is a d-variate copula, it is called *d-variate hierarchical Archimedean copula* (d-HAC) with the (tree) structure $(\mathcal{V}, \mathcal{E})$ and the labeling λ;
b) if $(\mathcal{V}, \mathcal{E})$ is binary, then $C_{(\mathcal{V},\mathcal{E},\lambda)}$ is called *binary*.

As this definition explicitly employs a tree structure $(\mathcal{V}, \mathcal{E})$, we can directly adopt necessary concepts from graph theory, which employs *ancestors* and *descendants* of a node, where the first refers to the set containing the parent of that node, the parent of this parent, etc., and the latter refers to the set containing the children of that node, the children of these children, etc. These concepts are shown in Table 1.

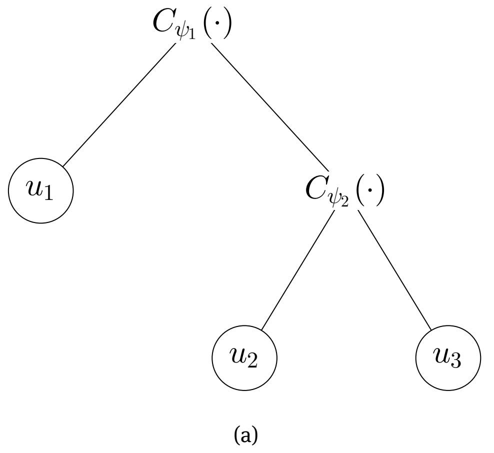
Table 1: The concepts adopted from graph theory.

concept	meaning
$\uparrow(v)$	the parent of v
$\Uparrow(v)$	the set of all ancestors of v
$\Downarrow(v)$	the set of all forks that are descendants of v
$\downarrow(v)$	the set of all leaves that are descendants of v, if v is a fork
$\downarrow(v)$	$\{v\}$, if v is a leave

For clarity, recall that $\wedge(v) \subset \Downarrow(v) \cup \downarrow(v)$ and also that $v \notin \Downarrow(v)$ and $v \notin \Uparrow(v)$.

Let us now illustrate the involved concepts on a simple example. Consider a HAC $C_{\psi_1,\psi_2}(u_1, u_2, u_3) = C_{\psi_1}(u_1, C_{\psi_2}(u_2, u_3))$ for two generators $\psi_1, \psi_2 \in \Psi$; its tree structure is depicted in Figure 1(a). For this representation, one can derive a tree with the same structure such as the one depicted in Figure 1(b) just by assigning different numbers to all of its nodes. For this tree, we have $\mathcal{V} = \{1, \ldots, 5\}$ and $\mathcal{E} = \{\{1,5\}, \{2,4\}, \{3,4\}, \{4,5\}\}$, which also imply that $f = 2$, the nodes $\{1,2,3\}$ are leaves and the nodes $\{4,5\}$ are forks. Let $\boldsymbol{u} \in \mathbb{I}^3$. Then $\boldsymbol{x^u} \in \mathbb{I}^5$, $\boldsymbol{x}^{\boldsymbol{u}}_1 = u_1$, $\boldsymbol{x}^{\boldsymbol{u}}_2 = u_2$, $\boldsymbol{x}^{\boldsymbol{u}}_3 = u_3$, $\boldsymbol{x}^{\boldsymbol{u}}_4 = C_{\lambda(4)}(\boldsymbol{x}^{\boldsymbol{u}}_{\{2,3\}}) = C_{\lambda(4)}(u_2, u_3)$ and analogously $\boldsymbol{x}^{\boldsymbol{u}}_5 = C_{\lambda(5)}(\boldsymbol{x}^{\boldsymbol{u}}_{\{1,4\}}) = C_{\lambda(5)}(u_1, C_{\lambda(4)}(u_2, u_3))$. Also note that $\uparrow(1) = \uparrow(4) = \Uparrow(1) = \Uparrow(4) = 5$, $\uparrow(2) = \uparrow(3) = 4$, $\Uparrow(2) = \Uparrow(3) = \{4,5\}$ for such a structure. Finally, $\Downarrow(5) = 4$, $\Downarrow(4) = \emptyset$, $\downarrow(5) = \{1,2,3\}$ and $\downarrow(4) = \{2,3\}$.

A necessary and sufficient condition for the function given by (4) to be a proper copula is not known. However, there are known several sufficient conditions. An early one based on complete monotonicity has been proposed in [11], see Theorem 4.4 therein, which is proven for a subclass of HACs called *fully nested Archimedean copulas*. Its generalization to all HACs has been proposed in [7]. A weaker sufficient condition



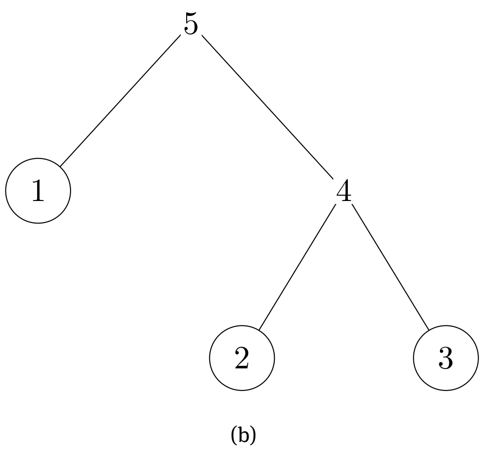


Figure 1: (a) A tree-like representation of a 3-HAC given by $C_{\psi_1,\psi_2}(u_1,u_2,u_3) = C_{\psi_1}(u_1, C_{\psi_2}(u_2,u_3))$. (b) A binary tree $(\mathcal{V},\mathcal{E})$, $\mathcal{V}=\{1,\dots,5\}$, $\mathcal{E}=\{\{1,5\},\{2,4\},\{3,4\},\{4,5\}\}$ corresponding to the structure of a 3-HAC $C_{(\mathcal{V},\mathcal{E},\lambda)}$.

based on d-monotonicity has appeared recently in [16] and is recalled in the following theorem. Note that $y_1 \circ y_2$ denotes the composition of two functions y_1 and y_2.

Theorem 1. [16] Let $(\mathcal{V},\mathcal{E})$ and λ be the tree and the labeling from Definition 1, respectively. Also let

$$h_m = \{y : [0,\infty] \to [0,\infty] \mid y(0)=0,\ y(\infty)=\infty,\ y' \text{ is } m\text{-monotone}\} \tag{5}$$

for all $m \geq 2$. If λ fulfills that

$$\begin{aligned} &1)\ \lambda(v) \text{ is } \#{\downarrow}(v)\text{-monotone, and} \\ &2)\ \lambda(v)^{-1} \circ \lambda(\tilde{v}) \in h_{\#\downarrow(\tilde{v})}, \end{aligned} \tag{6}$$

for all $v \in \{d+1,\dots,d+f\}$ and $\tilde{v} \in \wedge(v) \cap \{d+1,\dots,d+f\}$, then $C_{(\mathcal{V},\mathcal{E},\lambda)}$ is a copula.

The condition on λ stated in Theorem 1 will be called the *sufficient nesting condition* (*s.n.c.*). Also note that $h_2 \supset h_3 \supset \dots \supset h_m$ for all $m > 3$.

As has already been mentioned in the introduction, Kendall's tau is the most popular measure of concordance used in connection to estimation of HACs. In term of copulas, *Kendall's tau* of random vector $(U_1,U_2) \sim C$, where C is a bivariate copula, is given by

$$\tau(C) = 4 \int_{\mathbb{I}^2} C(u_1,u_2)dC(u_1,u_2) - 1. \tag{7}$$

Also, given a random vector $(U_1,\dots,U_d) \sim C$, where C is a d-variate copula, the matrix $(\tau_{ij}) \in \mathbb{I}^{d,d}$ such that for all $i,j \in \{1,\dots,d\}$ either

$$\begin{aligned} &1)\ \tau_{ij} = \tau(C_{ij}), \quad \text{if } i \neq j, \text{ or} \\ &2)\ \tau_{ij} = 1, \qquad \text{otherwise,} \end{aligned} \tag{8}$$

where C_{ij} is the (i,j)-th bivariate margin of C, is called the *Kendall correlation matrix* of $(U_1,\dots,U_d)$.

If C_ψ is a 2-AC and ψ is a twice continuously differentiable generator with $\psi(t) > 0$ for all $t \in [0,\infty)$, Kendall's tau can be represented as [8, p. 91], [13, p. 163]

$$\tau(\psi) = \tau(C_\psi) = 1 - 4\int_0^\infty t(\psi'(t))^2 dt = 1 + 4\int_0^1 \frac{\psi^{-1}(t)}{(\psi^{-1})'(t)}dt. \tag{9}$$

Due to this connection of generators and Kendall's tau, the restrictions on λ given by the s.n.c. also induce restrictions on λ in terms of Kendall's tau, as is stated in the following lemma.

Lemma 1. Let $C_{(\mathcal{V},\mathcal{E},\lambda)}$ be a d-HAC satisfying the s.n.c. stated in Theorem 1. Then

$$\tau(\lambda(v)) \leq \tau(\lambda(\tilde{\tilde{v}})) \tag{10}$$

for all $v \in \{d+1, \dots, d+f\}$ and $\tilde{\tilde{v}} \in \Downarrow(v)$.

Proof. Let us call a function $f : [0, \infty] \to \mathbb{R}$ *subadditive* if $f(x+y) \leq f(x)+f(y)$ for all $x, y \in [0, \infty)$. Let $v, \tilde{v} \in \mathcal{V}$ be two forks such that $\tilde{v} \in \wedge(v)$. Using Part 2) of the s.n.c. and the consideration that $\# \wedge (\tilde{v}) \geq 2$, see Definition 1, this implies that $(\lambda(v)^{-1} \circ \lambda(\tilde{v}))' \in h_2$ and thus $(\lambda(v)^{-1}(\lambda(\tilde{v})(t)))'' \leq 0$ for all $t \in (0, \infty)$, i.e., $\lambda(v)^{-1} \circ \lambda(\tilde{v})$ is concave on $(0, \infty)$. Further $\lambda(v)^{-1}(\lambda(\tilde{v})(0)) = 0$. Using Lemma 4.4.3 in [13, p. 136], it follows that $\lambda(v)^{-1} \circ \lambda(\tilde{v})$ is subadditive. With Theorem 4.4.2, this implies that $C_{\lambda(v)}(u_1, u_2) \leq C_{\lambda(\tilde{v})}(u_1, u_2)$ for all $u_1, u_2 \in \mathbb{I}$ and thus, as Kendall's tau reflects this concordance ordering, $\tau(\lambda(v)) \leq \tau(\lambda(\tilde{v}))$. The same reasoning can be done for a child of $\tilde{v}$, a child of a child of $\tilde{v}$, etc. Denoting such a descendant fork by $\tilde{\tilde{v}}$ establishes the proof. □

The condition given by (10) does not generally guarantee that a proper copula results. However, for a HAC with all generators from the same one-parametric Archimedean family, e.g., Ali-Mikhail-Haq, Clayton, Frank, Gumbel, Joe or many others, (10) combined with relatively weak assumptions turns to a sufficient nesting condition, as has been shown in Lemma 4.1 in [3]. In other words, under the assumption of the same family (e.g., one of the families mentioned above) for all generators in a HAC, one can aim to satisfy (10) instead of (6) in order to construct a proper copula, which is successfully applied in the construction of HAC estimators proposed in [3].

3 Structure determination

This section recalls the approach underlying the structure determination in `kt_kagg`. As has already been outlined in the introduction, `kt_kagg` merges the structure estimator from [2], which delivers a binary structure, with a collapsing procedure from [14], which, if needed, turns the binary structure to a non-binary one. The binary structure estimator from [2] stems from the approach introduced in [4], see Algorithm 2 therein, which, given a Kendall correlation matrix[1], applies agglomerative clustering in order to recover the underlying structure. Here, this approach is recalled in Algorithm 1. Note that a binary HAC $C_{(\mathcal{V},\mathcal{E},\lambda)}$ has always $d-1$ forks, which implies that $\mathcal{V} = \{1, \dots, 2d-1\}$. Also note that Algorithm 1 uses the following definition.

Definition 2. Let either $r = [0, \infty)$, or $r = [0, a]$, where $a > 0$. Any function $g : r^k \to r,\ k \in \mathbb{N}$, satisfying 1) $g(x, \dots, x) = x$ for all $x \in r$ and 2) g is exchangeable (i.e., $g(x_{p_1}, \dots, x_{p_k}) = g(x_1, \dots, x_k)$ for all $x_1, \dots, x_k \in r$ and all permutations $(p_1, \dots, p_k)$ of $(1, \dots, k)$) is called an *r-aggregation* function.

For example, the maximum, minimum or average restricted to $\mathbb{I}$ are $\mathbb{I}$-aggregation functions.

The goal of the algorithm is to build a HAC such that (10) is satisfied. I.e., the algorithm assigns forks corresponding to the highest values of τ (through $\tau(\lambda(\cdot))$) to the lowest levels of the branches in the structure. Ascending higher up in the tree, forks with lower values of τ are assigned.

1 Note that Algorithm 2 in [4] is stated more generally, i.e., it is possible to use it also with other concordance measures, not only with Kendall's tau.

Algorithm 1. Structure determination of a HAC based on Kendall's tau

```
Input:
  1) $(\tau_{ij})$ ... a Kendall correlation matrix
  2) $g$ ... an $\mathbb{I}$-aggregation function

The structure determination:
  1. $\hat{\mathcal{V}} = \{1, \dots, 2d-1\}$, $\hat{\mathcal{E}} = \emptyset$
  2. $\hat{\updownarrow}(k) = \{k\}$, $k = 1, \dots, d$
  3. $\mathcal{I} = \{1, \dots, d\}$,
  for $k = 1, \dots, d-1$ do
    4. find the pair of clusters (leaves) with the highest similarity according to $\tau$, i.e.,
       $(i, j) := \operatorname*{argmax}_{\tilde{i}<\tilde{j},\ \tilde{i}\in\mathcal{I},\ \tilde{j}\in\mathcal{I}} g((\tau_{\tilde{\tilde{i}}\tilde{\tilde{j}}})_{(\tilde{\tilde{i}},\tilde{\tilde{j}})\in\hat{\updownarrow}(\tilde{i})\times\hat{\updownarrow}(\tilde{j})})$
    5. join these two clusters, i.e.,
       $\hat{\updownarrow}(d+k) := \hat{\updownarrow}(i) \cup \hat{\updownarrow}(j)$
    6. remove the clusters $i$ and $j$ from the next considerations and substitute them by the cluster $d+k$, i.e.,
       $\mathcal{I} := \mathcal{I} \cup \{d+k\} \backslash \{i, j\}$
    7. create a part of the output structure by connecting the node $d+k$ with the nodes $i$ and $j$, i.e.,
       $\hat{\mathcal{E}} := \hat{\mathcal{E}} \cup \{\{i, d+k\}, \{j, d+k\}\}$ ..., i.e., $\wedge(d+k) := \{i, j\}$
  end for

Output:
  the binary tree $(\hat{\mathcal{V}}, \hat{\mathcal{E}})$
```

4 Justification of a tau-clustering-based approach

In [4], the approach recalled in the previous section was theoretically justified only for $d = 4$. In this section, as the main result of this work, a theorem that generalizes this justification to an arbitrary dimension is proposed. Before its statement, three auxiliary lemmas are provided. Note that, as Algorithm 1 serves for delivering binary structures, the considerations below are also mostly restricted to such structures. However, the consequences of the main theorem are discussed also for the non-binary case.

Definition 3. [2] Let $C_{(\mathcal{V},\mathcal{E},\lambda)}$ be a d-HAC and i, j two different leaves in $\mathcal{V}$ such that neither i nor j is the root. We call *youngest common ancestor* of i, j (denoted by $\Uparrow_Y(i, j)$) the fork $m \in \mathcal{V}$ for which $m \in \Uparrow(i) \cap \Uparrow(j)$ and $\Uparrow(i) \cap \Uparrow(j) \cap \Downarrow(m) = \emptyset$.

Remark 1. Given a d-HAC $C_{(\mathcal{V},\mathcal{E},\lambda)}$, as $(\mathcal{V}, \mathcal{E})$ is a tree, if $i, j \in \mathcal{V}$ are two different nodes such that neither i nor j is the root, it follows that the set $\Uparrow(i) \cap \Uparrow(j)$ always contains the root. This thus implies the existence of $\Uparrow_Y(i, j)$ for such a pair of nodes.

Remark 2. Observe that if $C_{(\mathcal{V},\mathcal{E},\lambda)}$ is a binary d-HAC, m is a fork in $\mathcal{V}$, $\wedge(m) = \{l, r\}$, $i \in \downarrow(l) \cup \Downarrow(l)$ and $j \in \downarrow(r) \cup \Downarrow(r)$, then $\Uparrow_Y(i, j) = m$.

Lemma 2. Let $C_{(\mathcal{V},\mathcal{E},\lambda)}$ be a binary d-HAC and $\{i_1, \dots, i_k\} \subset \mathcal{V}$, $k \geq 2$ be such that the sets $\downarrow(i_1), \dots, \downarrow(i_k)$ are a disjoint decomposition of $\{1, \dots, d\}$. Then for each fork $m \in \Uparrow(i_1) \cup \dots \cup \Uparrow(i_k)$, there exists a pair (i, j), $i < j$, $i, j \in \{i_1, \dots, i_k\}$ such that $\Uparrow_Y(i, j) = m$.

Proof. Assume that $m \in \Uparrow(i_1) \cup \dots \cup \Uparrow(i_k)$, $\{l, r\} = \wedge(m)$, $\tilde{i} \in \downarrow(l)$ and $\tilde{j} \in \downarrow(r)$, i.e, according to Remark 2, $\Uparrow_Y(\tilde{i}, \tilde{j}) = m$. Since $(\mathcal{V}, \mathcal{E})$ is a binary tree and the sets $\downarrow(i_1), \dots, \downarrow(i_k)$ are a disjoint decomposition of $\{1, \dots, d\}$, there exists either a fork $i \in \{i_1, \dots, i_k\}$ such that $\tilde{i} \in \downarrow(i)$ or a leaf $i \in \{i_1, \dots, i_k\}$ such that $\tilde{i} = i$. Analogously, there exists either a fork $j \in \{i_1, \ldots, i_k\}$ such that $\tilde{j} \in \downarrow(j)$ or a leaf $j \in \{i_1, \ldots, i_k\}$ such that $\tilde{j} = j$. Also, as $\tilde{i} \in \downarrow(l)$, it implies that $i \in \downarrow(l) \cup \Downarrow(l)$. Analogously, $j \in \downarrow(r) \cup \Downarrow(r)$. Using Remark 2, the proof is established. □

Lemma 3. [2] Let $C_{(\mathcal{V},\mathcal{E},\lambda)}$ be a binary d-HAC. Then

$$C_{(\mathcal{V},\mathcal{E},\lambda)}(1, \ldots, 1, u_i, 1, \ldots, 1, u_j, 1, \ldots, 1) = C_{\lambda(\Uparrow_{\mathcal{V}}(i,j))}(u_i, u_j),\ 1 \le i < j \le d \tag{11}$$

for all $(u_i, u_j) \in \mathbb{I}^2$.

Hence for all $i, j \in \{1, \ldots, d\}$ such that $i \neq j$, the random vector (U_i, U_j) is distributed according to the 2-AC $C_{\lambda(\Uparrow_{\mathcal{V}}(i,j))}$, where $(U_1, \ldots, U_d) \sim C_{(\mathcal{V},\mathcal{E},\lambda)}$. Note that Lemma 3 is actually Proposition 3 introduced in [2].

Lemma 4. Let $C_{(\mathcal{V},\mathcal{E},\lambda)}$ be a binary d-HAC, $(U_1, \ldots, U_d) \sim C_{(\mathcal{V},\mathcal{E},\lambda)}$ and (τ_{ij}) be the Kendall correlation matrix of $(U_1, \ldots, U_d)$. Then

$$\tau_{ij} = \tau(\lambda(\Uparrow_{\mathcal{V}}(i, j))) \tag{12}$$

for all leaves i and j such that $i \neq j$.

Proof. This directly follows from Lemma 3, i.e., as $(U_i, U_j) \sim C_{\lambda(\Uparrow_{\mathcal{V}}(i,j))}$, it is $\tau_{ij} = \tau(C_{\lambda(\Uparrow_{\mathcal{V}}(i,j))}) = \tau(\lambda(\Uparrow_{\mathcal{V}}(i, j)))$. □

Now follows the main theorem of this work.

Theorem 2. Let $C_{(\mathcal{V},\mathcal{E},\lambda)}$ be a binary d-HAC satisfying the s.n.c. given by (6) and

$$\tau(\lambda(2d - 1)) < \tau(\lambda(2d - 2)) < \ldots < \tau(\lambda(d + 1)), \tag{13}$$

$(U_1, \ldots, U_d) \sim C_{(\mathcal{V},\mathcal{E},\lambda)}$, and (τ_{ij}) be the Kendall correlation matrix of $(U_1, \ldots, U_d)$. Then, given the input (τ_{ij}) and any $\mathbb{I}$-aggregation fuction g, Algorithm 1 returns $(\mathcal{V}, \mathcal{E})$.

Proof. First consider that, if one knows the children of all forks in a HAC structure (in a tree $(\mathcal{V}, \mathcal{E})$), then one knows the whole HAC structure, i.e., the sets $\wedge(d + 1), \ldots, \wedge(2d - 1)$ determine $(\mathcal{V}, \mathcal{E})$. Hence, in the proof, instead of showing directly that $(\mathcal{V}, \mathcal{E}) = (\hat{\mathcal{V}}, \hat{\mathcal{E}})$, we show that $\hat{\wedge}(k) = \wedge(k)$ for all $k \in \{d+1, \ldots, 2d-1\}$, where $\hat{\wedge}(k)$ denotes the children of a fork k in the structure $(\hat{\mathcal{V}}, \hat{\mathcal{E}})$.

The proof is performed by induction according to k. Let $k = 1$. Also, let $\mathcal{I} = \{1, \ldots, d\}$ and

$$(i, j) = \underset{\tilde{i}<\tilde{j},\ \tilde{i}\in\mathcal{I},\ \tilde{j}\in\mathcal{I}}{\operatorname{argmax}}\ g((\tau_{\tilde{\tilde{i}}\tilde{\tilde{j}}})_{(\tilde{\tilde{i}},\tilde{\tilde{j}})\in\downarrow(\tilde{i})\times\downarrow(\tilde{j})}). \tag{14}$$

As $\mathcal{I} = \{1, \ldots, d\}$, it is (by the definition, see Section 2) $\downarrow(\tilde{i}) = \{\tilde{i}\}$ and $\downarrow(\tilde{j}) = \{\tilde{j}\}$ for all $\tilde{i}, \tilde{j} \in \mathcal{I}$. Hence, given $\tilde{i}, \tilde{j} \in \mathcal{I}$, it follows that $g((\tau_{\tilde{\tilde{i}}\tilde{\tilde{j}}})_{(\tilde{\tilde{i}},\tilde{\tilde{j}})\in\downarrow(\tilde{i})\times\downarrow(\tilde{j})}) = g(\tau_{\tilde{i}\tilde{j}}) = \tau_{\tilde{i}\tilde{j}}$; for the latter equality, see Definition 2. Hence, (14) simplifies to

$$(i, j) = \underset{\tilde{i}<\tilde{j},\ \tilde{i}\in\mathcal{I},\ \tilde{j}\in\mathcal{I}}{\operatorname{argmax}}\ \tau_{\tilde{i}\tilde{j}}. \tag{15}$$

Note that the argument corresponding to the maximum is unique, which is addressed at the end of this induction step. Now, rewrite (15) to

$$\tau_{ij} = \max_{\tilde{i}<\tilde{j},\ \tilde{i}\in\mathcal{I},\ \tilde{j}\in\mathcal{I}} \tau_{\tilde{i}\tilde{j}}. \tag{16}$$

As, according to Lemma 4, $\tau_{\tilde{i}\tilde{j}} = \tau(\lambda(\Uparrow_{\mathcal{V}}(\tilde{i}, \tilde{j})))$, $\tilde{i} < \tilde{j}$, $\tilde{i}, \tilde{j} \in \mathcal{I}$, (16) turns to

$$\max_{\tilde{i}<\tilde{j},\ \tilde{i}\in\mathcal{I},\ \tilde{j}\in\mathcal{I}} \tau_{\tilde{i}\tilde{j}} = \max_{\tilde{i}<\tilde{j},\ \tilde{i}\in\mathcal{I},\ \tilde{j}\in\mathcal{I}} \tau(\lambda(\Uparrow_{\mathcal{V}}(\tilde{i}, \tilde{j}))). \tag{17}$$

According to Remark 1, for each pair $(\tilde{i},\tilde{j})$, $\tilde{i}<\tilde{j}$, $\tilde{i},\tilde{j}\in\mathcal{I}$ there exists a fork $v\in\{d+1,\dots,2d-1\}$ such that $\Uparrow_{\mathcal{Y}}(\tilde{i},\tilde{j})=v$ and, vice versa, according to Lemma 2, for each fork $v\in\{d+1,\dots,2d-1\}$, there exists a pair $(\tilde{i},\tilde{j})$ such that $\tilde{i}<\tilde{j}$, $\tilde{i},\tilde{j}\in\mathcal{I}$ and $\Uparrow_{\mathcal{Y}}(\tilde{i},\tilde{j})=v$. This in turn implies that

$$\max_{\tilde{i}<\tilde{j},\ \tilde{i}\in\mathcal{I},\ \tilde{j}\in\mathcal{I}}\tau(\lambda(\Uparrow_{\mathcal{Y}}(\tilde{i},\tilde{j})))=\max_{v\in\{d+1,\dots,2d-1\}}\tau(\lambda(v)). \tag{18}$$

According to (13),

$$\max_{v\in\{d+1,\dots,2d-1\}}\tau(\lambda(v))=\tau(\lambda(d+1)), \tag{19}$$

and thus $\tau_{ij}=\tau(\lambda(d+1))$. But also, as follows from Lemma 4, $\tau_{ij}=\tau(\lambda(\Uparrow_{\mathcal{Y}}(i,j)))$. Considering (13), it implies that $\tau(\lambda(m))<\tau(\lambda(d+1))$ for any fork m such that $m\neq d+1$. It is thus clear that $d+1=\Uparrow_{\mathcal{Y}}(i,j)$.

However, it is still not clear if $d+1=\uparrow(i)=\uparrow(j)$, i.e., if

$$\{i,j\}=\wedge(d+1). \tag{20}$$

But, if (20) would not hold, then it would exist a fork $m\in\{d+2,\dots,2d-1\}$, such that $\uparrow(i)=m$ (or analogously for $\uparrow(j)=m$). This would imply that $m\in\Downarrow(d+1)$ and thus, according to Lemma 1, $\tau(\lambda(d+1))\le\tau(\lambda(m))$. However, this contradicts (13).

Now discuss the uniqueness of the maximum in (15). Assume that there exists a pair $\{i_0,j_0\}\subset\mathcal{I}$ such that $i_0\notin\{i,j\}$ and

$$(i_0,j_0)=\operatorname*{argmax}_{\tilde{i}<\tilde{j},\ \tilde{i}\in\mathcal{I},\ \tilde{j}\in\mathcal{I}}\tau_{\tilde{\tilde{i}}\tilde{\tilde{j}}}. \tag{21}$$

With the same argument as above, it would imply that $\uparrow(i_0)=d+1$ (or analogously $\uparrow(j_0)=d+1$ in the case that the assumption would be $j_0\notin\{i,j\}$). However, this contradicts the fact that $(\mathcal{V},\mathcal{E})$ is binary.

Finally, considering that (14) is Step 4 of Algorithm 1, this implies that

$$\hat{\wedge}(d+1)=\wedge(d+1), \tag{22}$$

see also Step 7. Now, update $\mathcal{I}:=\mathcal{I}\cup\{d+1\}\backslash\{i,j\}$, define that $\{\tilde{i}_1,\dots,\tilde{i}_{d-1}\}=\mathcal{I}$, and observe that $\downarrow(\tilde{i}_1)\cup\ldots\cup\downarrow(\tilde{i}_{d-1})=\{1,\dots,d\}$ and $\Uparrow(\tilde{i}_1)\cup\ldots\cup\Uparrow(\tilde{i}_{d-1})=\{d+2,\dots,2d-1\}$, where the latter follows from the consideration that $\Uparrow(1)\cup\ldots\cup\Uparrow(d)=\{d+1,\dots,2d-1\}$ and the fact that $d+1$ is the parent of the leaves i and j and thus cannot be an ancestor of any node from $\mathcal{I}$. Also observe that the sets $\downarrow(\tilde{i}_1),\dots,\downarrow(\tilde{i}_{d-1})$ are disjoint and thus they are a disjoint decomposition of $\{1,\dots,d\}$.

Now the induction step for $k\in\{2,\dots,d-1\}$. Assume that $\hat{\wedge}(d+m)=\wedge(d+m)$ for all $m\in\{1,\dots,k-1\}$ and that $\mathcal{I}=\{i_1,\dots,i_{d-k+1}\}$ such that $i_1,\dots,i_{d-k+1}<d+k$, the sets $\downarrow(i_1),\dots,\downarrow(i_{d-k+1})$ are a disjoint decomposition of $\{1,\dots,d\}$ and $\Uparrow(i_1)\cup\ldots\cup\Uparrow(i_{d-k+1})=\{d+k,\dots,2d-1\}$. Again, let

$$(i,j)=\operatorname*{argmax}_{\tilde{i}<\tilde{j},\ \tilde{i}\in\mathcal{I},\ \tilde{j}\in\mathcal{I}}g((\tau_{\tilde{\tilde{i}}\tilde{\tilde{j}}})_{(\tilde{\tilde{i}},\tilde{\tilde{j}})\in\downarrow(\tilde{i})\times\downarrow(\tilde{j})}). \tag{23}$$

Since the sets $\downarrow(\tilde{i})$ and $\downarrow(\tilde{j})$ are disjoint for any pair $(\tilde{i},\tilde{j})$ in the domain of maximization and thus neither $\tilde{i}\in\Downarrow(\tilde{j})$ nor $\tilde{j}\in\Downarrow(\tilde{i})$, given $\tilde{i},\tilde{j}\in\mathcal{I}$, $\tilde{i}<\tilde{j}$, it holds that $\Uparrow_{\mathcal{Y}}(\tilde{\tilde{i}},\tilde{\tilde{j}})=\Uparrow_{\mathcal{Y}}(\tilde{i},\tilde{j})$ for all $(\tilde{\tilde{i}},\tilde{\tilde{j}})\in\downarrow(\tilde{i})\times\downarrow(\tilde{j})$, which follows from the consideration that $(\mathcal{V},\mathcal{E})$ is a binary tree. Hence, it follows that, using Lemma 4, $\tau_{\tilde{\tilde{i}}\tilde{\tilde{j}}}=\tau(\lambda(\Uparrow_{\mathcal{Y}}(\tilde{\tilde{i}},\tilde{\tilde{j}})))=\tau(\lambda(\Uparrow_{\mathcal{Y}}(\tilde{i},\tilde{j})))$ for all $(\tilde{\tilde{i}},\tilde{\tilde{j}})\in\downarrow(\tilde{i})\times\downarrow(\tilde{j})$ and thus $g((\tau_{\tilde{\tilde{i}}\tilde{\tilde{j}}})_{(\tilde{\tilde{i}},\tilde{\tilde{j}})\in\downarrow(\tilde{i})\times\downarrow(\tilde{j})})=g(\tau(\lambda(\Uparrow_{\mathcal{Y}}(\tilde{i},\tilde{j}))),\dots,\tau(\lambda(\Uparrow_{\mathcal{Y}}(\tilde{i},\tilde{j}))))$. Using Part 1) of Definition 2, we get $g(\tau(\lambda(\Uparrow_{\mathcal{Y}}(\tilde{i},\tilde{j}))),\dots,\tau(\lambda(\Uparrow_{\mathcal{Y}}(\tilde{i},\tilde{j}))))=\tau(\lambda(\Uparrow_{\mathcal{Y}}(\tilde{i},\tilde{j})))$. Hence, (23) turns to

$$(i,j)=\operatorname*{argmax}_{\tilde{i}<\tilde{j},\ \tilde{i}\in\mathcal{I},\ \tilde{j}\in\mathcal{I}}\tau(\lambda(\Uparrow_{\mathcal{Y}}(\tilde{i},\tilde{j}))). \tag{24}$$

Note that the argument corresponding to the maximum is unique for the same reason as in the induction step $k=1$. Again, rewrite (24) to

$$\tau(\lambda(\Uparrow_{\mathcal{Y}}(i,j)))=\max_{\tilde{i}<\tilde{j},\ \tilde{i}\in\mathcal{I},\ \tilde{j}\in\mathcal{I}}\tau(\lambda(\Uparrow_{\mathcal{Y}}(\tilde{i},\tilde{j}))). \tag{25}$$

Using the assumption that $\downarrow(i_1), \dots, \downarrow(i_{d-k+1})$ are a disjoint decomposition of $\{1, \dots, d\}$, Remark 1 implies that for each pair $(\tilde{i}, \tilde{j})$ such that $\tilde{i} < \tilde{j}$ and $\tilde{i}, \tilde{j} \in \mathcal{I}$ there exist a fork $v \in \Uparrow(i_1) \cup \dots \cup \Uparrow(i_{d-k+1})$ such that $v = \Uparrow_Y(\tilde{i}, \tilde{j})$ and, vice versa, according to Lemma 2, for each fork $v \in \Uparrow(i_1) \cup \dots \cup \Uparrow(i_{d-k+1})$, there exists a pair $(\tilde{i}, \tilde{j})$ such that $\tilde{i} < \tilde{j}$, $\tilde{i}, \tilde{j} \in \mathcal{I}$ and $\Uparrow_Y(\tilde{i}, \tilde{j}) = v$. Using the assumption that $\Uparrow(i_1) \cup \dots \cup \Uparrow(i_{d-k+1}) = \{d+k, \dots, 2d-1\}$, it follows that

$$\max_{\tilde{i}<\tilde{j},\, \tilde{i}\in\mathcal{I},\, \tilde{j}\in\mathcal{I}} \tau(\lambda(\Uparrow_Y(\tilde{i}, \tilde{j}))) = \max_{v\in\{d+k,\dots,2d-1\}} \tau(\lambda(v)), \tag{26}$$

and in turn, using (13),

$$\max_{v\in\{d+k,\dots,2d-1\}} \tau(\lambda(v)) = \tau(\lambda(d+k)). \tag{27}$$

As the maximum is unique, it holds that $d+k = \Uparrow_Y(i, j)$. But also $\uparrow(i) = \uparrow(j) = d+k$, because, if that would not hold, then it would exist a fork $m \in \{d+k+1, \dots, 2d-1\}$ such that $\uparrow(i) = m$ (or analogously for $\uparrow(j) = m$). This would imply that $m \in \Downarrow(d+k)$ and thus $\tau(\lambda(d+k)) \le \tau(\lambda(m))$ according to Lemma 1. However, this contradicts (13).

Finally, considering that (14) is Step 4 of Algorithm 1, this implies that

$$\hat{\wedge}(d+k) = \wedge(d+k), \tag{28}$$

see also Step 7. Now, perform the update $\mathcal{I} := \mathcal{I} \cup \{d+k\} \backslash \{i, j\}$, define that $\{\tilde{i}_1, \dots, \tilde{i}_{d-k}\} = \mathcal{I}$, and observe that $\downarrow(\tilde{i}_1) \cup \dots \cup \downarrow(\tilde{i}_{d-k}) = \{1, \dots, d\}$ and that the sets $\downarrow(\tilde{i}_1), \dots, \downarrow(\tilde{i}_{d-k})$ are disjoint and thus they are a disjoint decomposition of $\{1, \dots, d\}$. Also observe that $\tilde{i}_1, \dots, \tilde{i}_{d-k} < d+k+1$ and $\Uparrow(\tilde{i}_1) \cup \dots \cup \Uparrow(\tilde{i}_{d-k}) = \{d+k+1, \dots, 2d-1\}$, where the latter follows from the assumption that $\Uparrow(i_1) \cup \dots \cup \Uparrow(i_{d-k+1}) = \{d+k, \dots, 2d-1\}$ and the fact that $d+k$ is the parent of the nodes i and j and thus cannot be an ancestor of any node from $\mathcal{I}$, which follows from the assumption that the sets $\downarrow(i_1), \dots, \downarrow(i_{d-k+1})$ are a disjoint decomposition of $\{1, \dots, d\}$.
□

5 Discussion

It follows from Theorem 2 that the choice of the $\mathbb{I}$-aggregation function g does not influence the output of Algorithm 1. However note that if Algorithm 1 is transformed to an algorithm for HAC estimation, as, e.g, has been shown in [2], see Section 4 therein, the choice of g becomes essential for the fit of the HAC estimates, as is also experimentally shown in the cited article.

To recover the structure of a HAC, Algorithm 1 relies on the Kendall's correlation coefficient, which reflects the popularity of this coefficient in practical applications concerning ACs and HACs. However, it is clear from the proof of Theorem 2 that any other measure of concordance, e.g, Spearman's Rho, can be used instead provided an analogue of Lemma 1 for such a measure is available.

Now, let us consider the assumption given by (13). Assume that $C_{(\mathcal{V},\mathcal{E}_1,\lambda_1)}$ and $C_{(\mathcal{V},\mathcal{E}_2,\lambda_2)}$ depicted in Figure 2 are 4-HACs. Observe that $C_{(\mathcal{V},\mathcal{E}_2,\lambda_2)}$ results from $C_{(\mathcal{V},\mathcal{E}_1,\lambda_1)}$ if the forks 5 and 6 in $(\mathcal{V}, \mathcal{E}_1, \lambda_1)$ are *permuted* in $(\mathcal{V}, \mathcal{E}_2, \lambda_2)$, i.e.,

$$\lambda_2(5) = \lambda_1(6) \text{ and } \lambda_2(6) = \lambda_1(5), \tag{29}$$

and for the rest of the forks the labeling is the same (i.e., $\lambda_2(7) = \lambda_1(7)$), and the structure $(\mathcal{V}_2, \mathcal{E}_2)$ relates to $(\mathcal{V}_1, \mathcal{E}_1)$ as depicted in Figure 2. This can be seen from the fact that $C_{(\mathcal{V},\mathcal{E}_1,\lambda_1)}(\boldsymbol{u}) = C_{\lambda_1(7)}(u_1, C_{\lambda_1(6)}(u_2, C_{\lambda_1(5)}(u_3, u_4))) = C_{\lambda_2(7)}(u_1, C_{\lambda_2(5)}(u_2, C_{\lambda_2(6)}(u_3, u_4))) = C_{(\mathcal{V},\mathcal{E}_2,\lambda_2)}(\boldsymbol{u})$ holds for all $\boldsymbol{u} \in \mathbb{I}^4$, where the second equality follows from (29).

As follows from Lemma 1, (13) cannot be satisfied for $C_{(\mathcal{V},\mathcal{E}_2,\lambda_2)}$. However, one can use instead of it the same copula function $C_{(\mathcal{V},\mathcal{E}_1,\lambda_1)}$ constructed in the way described above. Generally, given a d-HAC $C_{(\mathcal{V},\mathcal{E}_2,\lambda_2)}$ satisfying the s.n.c. but violating (13), one can in the same way as described above construct a d-HAC $C_{(\mathcal{V},\mathcal{E}_1,\lambda_1)}$

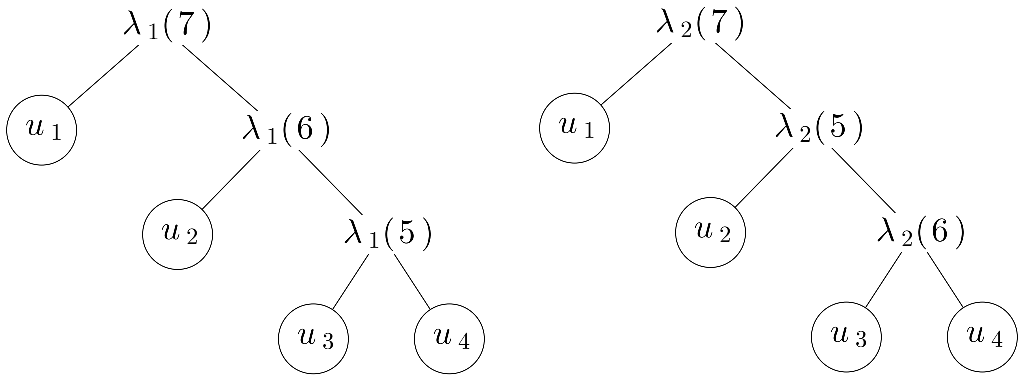


Figure 2: Two 4-HACs $C_{(\mathcal{V},\mathcal{E}_1,\lambda_1)}$ and $C_{(\mathcal{V},\mathcal{E}_2,\lambda_2)}$ such that $(\mathcal{V}, \mathcal{E}_2, \lambda_2)$ results from $(\mathcal{V}, \mathcal{E}_1, \lambda_1)$, if the forks 5 and 6 in $(\mathcal{V}, \mathcal{E}_1, \lambda_1)$ are *permuted*, i.e, $\lambda_2(5) = \lambda_1(6)$, $\lambda_2(6) = \lambda_1(5)$ and $\lambda_2(7) = \lambda_1(7)$), and $\mathcal{E}_1 = \{\{7, 1\}, \{7, 6\}, \{6, 2\}, \{6, 5\}, \{5, 3\}, \{5, 4\}\}$ and $\mathcal{E}_2 = \{\{7, 1\}, \{7, 5\}, \{5, 2\}, \{5, 6\}, \{6, 3\}, \{6, 4\}\}$.

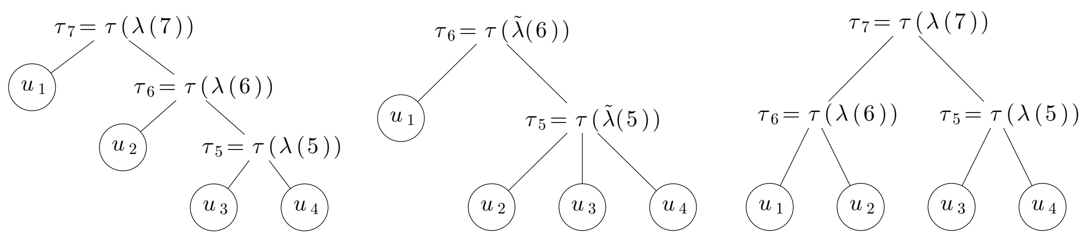
such that (13) is satisfied. This can be done just by assigning the fork $d + 1$ to the generator corresponding to the highest Kendall's tau, $d + 2$ to the generator corresponding to the second highest Kendall's tau, etc. This implies that the assumption (13) can be weakened to

$$\tau(\lambda(i)) \neq \tau(\lambda(j)) \text{ for all } i \neq j,\ i, j \in \{d + 1, \dots, 2d - 1\}. \tag{30}$$

Finally note that, if $C_{(\mathcal{V},\mathcal{E},\lambda)}$ satisfies both the s.n.c. and (30), it holds that $\tau(\lambda(2d - 1)) < \tau(\lambda(k))$ for all $k \in \{d + 1, \dots, 2d - 2\}$, i.e., it is never necessary to permute the root with some other fork in order to construct a HAC satisfying (13) in the sense described above. This follows from the fact that, assuming $\{l, r\} = \wedge(2d - 1)$, according to Lemma 1, $\tau(\lambda(2d-1)) \le \tau(\lambda_2(l))$ and $\tau(\lambda(2d-1)) \le \tau(\lambda(r))$, and using (30), it turns to $\tau(\lambda(2d-1)) < \tau(\lambda(l))$ and $\tau(\lambda(2d - 1)) < \tau(\lambda(r))$. The inequality for the rest of the forks follows again from Lemma 1 and the previous consideration.

Now, let us concern how Algorithm 1 behaves when (30) is not satisfied. The two following cases can be distinguished:

1. There is a parent-child pair of forks such that their corresponding values of Kendall's tau are equal. E.g., let $C_{(\mathcal{V},\mathcal{E},\lambda)}$ depicted in Figure 3(a) be a 4-HAC such the s.n.c. is satisfied and $\tau_5 = \tau_6 > \tau_7$. As addressed in the proof of Theorem 2, for $k = 1$, in Step 4, the algorithm searches for the pair of leaves corresponding to the largest tau. Under our assumptions here, however, there are three pairs corresponding to the maximum, see the Kendall correlation matrix in Figure 3(d). Also consider that the algorithm does not explicitly state any specific procedure for choosing the pair (i, j) in such a case. Hence, a common approach is to take one of the pairs arbitrarily, which of course generally *does not* lead to the true structure. But, consider the implications of our assumptions.
Assuming $\lambda(5)$ and $\lambda(6)$ to belong to a one-parametric family of generators (or possibly to two different one-parametric families of generators), which is a common assumption in many applications of HACs, for a lot of families, it follows that $\tau_5 = \tau_6$ implies that $C_{\lambda(5)} = C_{\lambda(6)}$, e.g., cf. Tables 1, 2 and 3 in [3]. Hence,


$\tau_7 = \tau(\lambda(7))$
u_1
$\tau_6 = \tau(\lambda(6))$
u_2
$\tau_5 = \tau(\lambda(5))$
u_3
u_4
(a)
$\tau_6 = \tau(\tilde{\lambda}(6))$
u_1
$\tau_5 = \tau(\tilde{\lambda}(5))$
u_2
u_3
u_4
(b)
$\tau_7 = \tau(\lambda(7))$
$\tau_6 = \tau(\lambda(6))$
$\tau_5 = \tau(\lambda(5))$
u_1
u_2
u_3
u_4
(c)

$$\begin{pmatrix} \cdot & \tau_7 & \tau_7 & \tau_7 \\ \cdot & \cdot & \tau_6 & \tau_6 \\ \cdot & \cdot & \cdot & \tau_5 \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \qquad \begin{pmatrix} \cdot & \tau_6 & \tau_6 & \tau_6 \\ \cdot & \cdot & \tau_5 & \tau_5 \\ \cdot & \cdot & \cdot & \tau_5 \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \qquad \begin{pmatrix} \cdot & \tau_6 & \tau_7 & \tau_7 \\ \cdot & \cdot & \tau_7 & \tau_7 \\ \cdot & \cdot & \cdot & \tau_5 \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

(d) (e) (f)

Figure 3: Three 4-HACs and their corresponding Kendall correlation matrices.

applying this assumption, it follows that

$$\begin{aligned}
&C_{(\mathcal{V},\mathcal{E},\lambda)}(u_1, u_2, u_3, u_4) = \\
&C_{\lambda(7)}(u_1, C_{\lambda(6)}(u_2, C_{\lambda(5)}(u_3, u_4))) = \\
&C_{\lambda(7)}(u_1, C_{\lambda(5)}(u_2, C_{\lambda(5)}(u_3, u_4))) = \\
&C_{\lambda(7)}(u_1, C_{\lambda(5)}(u_2, \lambda(5)(\lambda(5)^{-1}(u_3) + \lambda(5)^{-1}(u_4)))) = \\
&C_{\lambda(7)}(u_1, \lambda(5)(\lambda(5)^{-1}(u_2) + \lambda(5)^{-1}(\lambda(5)(\lambda(5)^{-1}(u_3) + \lambda(5)^{-1}(u_4))))) = \\
&C_{\lambda(7)}(u_1, \lambda(5)(\lambda(5)^{-1}(u_2) + (\lambda(5)^{-1}(u_3) + \lambda(5)^{-1}(u_4)))) = \\
&C_{\lambda(7)}(u_1, C_{\lambda(5)}(u_2, u_3, u_4)). \qquad (31)
\end{aligned}$$

But also

$$C_{\lambda(7)}(u_1, C_{\lambda(5)}(u_2, u_3, u_4)) = C_{\lambda(7)}(u_1, C_{\lambda(5)}(u_3, C_{\lambda(5)}(u_2, u_4))) = C_{\lambda(7)}(u_1, C_{\lambda(5)}(u_4, C_{\lambda(5)}(u_2, u_3))), \quad (32)$$

which follows from the consideration that $C_{\lambda(5)}$ is an AC. Looking at Definition 1, one can thus easily see that if $\lambda(5) = \lambda(6)$, one can use three different binary structures to get the same copula function (with $\{3, 4\}$, $\{2, 4\}$ or $\{2, 3\}$ at the lowest level, respectively). In other words, these three structures are *unidentifiable* based on such a copula function.
However, denoting $\lambda(7)$ by $\tilde{\lambda}(6)$ and $\lambda(5)$ by $\tilde{\lambda}(5)$, one obtains the same copula function but with a different structure that is depicted in Figure 3(b), i.e., if there is a parent-child pair of forks such that their corresponding values of Kendall's tau are equal, one can derive the *same* copula function but with a *non-binary* structure in the way mentioned above. What about determining this structure instead of the unidentifiable binary one?
Let us get back to the binary 4-HAC $C_{(\mathcal{V},\mathcal{E},\lambda)}$ in Figure 3(a) and to the idea where (for $k = 1$) one chooses arbitrarily the pair of nodes that will be assigned to the lowest level. It follows from Algorithm 1 that, for $k = 2$, the chosen pair in Step 4 will be $(m, 5)$, where m is the leaf that was not chosen in the loop for $k = 1$ out of $\{2, 3, 4\}$. Just to be complete, for $k = 3$, the chosen pair will be $(1, 6)$. Now, if we apply a collapsing strategy that collapses two parent-child nodes into one if the corresponding values of Kendall's tau are equal, the nodes 6 and 5 collapse to a new node, resulting in a non-binary structure described above, which is depicted in Figure 3(b). In other words, such an approach leads in this case to the true non-

binary structure. Proving that Algorithm 1 together with such a collapsing strategy leads generally to the true non-binary structure remains still an open problem.

2. There is a pair of forks such that their corresponding values of Kendall's tau are equal but these forks appear at different branches. E.g., let $C_{(\mathcal{V},\mathcal{E},\lambda)}$ depicted in Figure 3(c) be a 4-HAC such the s.n.c. is satisfied and $\tau_5 = \tau_6$. In this case, again, for $k = 1$, one can arbitrarily choose $(i, j) = (1, 2)$ or $(i, j) = (3, 4)$ in Step 4, see the Kendall correlation matrix in Figure 3(f). Let choose the first possibility. For $k = 2$, it is easy to see that $(i, j) = (3, 4)$, and for $k = 3$, $(i, j) = (5, 6)$, i.e., the true structure has been obtained. One can also easily see that if $(i, j) = (3, 4)$ is chosen for $k = 1$ in Step 4, the result is the same. Proving that Algorithm 1 leads generally to the true structure for this case remains still an open problem.

6 Conclusion

This work considered an existing approach to structure determination of HACs based on agglomerative clustering and the Kendall correlation matrix. The approach was first recalled including a detailed example, and then, as the main results of this work, a theorem showing that it leads to the true structure was proven. The assumptions of the theorem were discussed and two open problems for further research were identified.

Acknowledgement: This research was supported by the Ministry of Education, Youth and Sports of the Czech Republic within the Institutional Support for Long-term Development of a Research Organization in 2016 and partially supported by the Czech Science Foundation (GAČR) grant 17-01251.

References

[1] Górecki, J., M. Hofert, and M. Holeňa (2014). On the consistency of an estimator for hierarchical Archimedean copulas. In J. Talašová, J. Stoklasa, and T. Talášek (Eds.), *32nd International Conference on Mathematical Methods in Economics*, pp. 239–244. Palacký University, Olomouc.

[2] Górecki, J., M. Hofert, and M. Holeňa (2016a). An approach to structure determination and estimation of hierarchical Archimedean copulas and its application to bayesian classification. *J. Intell. Inf. Syst. 46*(1), 21–59.

[3] Górecki, J., M. Hofert, and M. Holeňa (2016b). On structure, family and parameter estimation of hierarchical Archimedean copulas. Available at https://arxiv.org/abs/1611.09225.

[4] Górecki, J. and M. Holeňa (2013). An alternative approach to the structure determination of hierarchical Archimedean copulas. *Proceedings of the 31st International Conference on Mathematical Methods in Economics (MME 2013), Jihlava*, pp. 201 – 206.

[5] Górecki, J. and M. Holeňa (2014). Structure determination and estimation of hierarchical Archimedean copulas based on Kendall correlation matrix. In A. Appice, M. Ceci, C. Loglisci, G. Manco, E. Masciari, and Z. W. Ras (Eds.), *New Frontiers in Mining Complex Patterns*, pp. 132–147. Springer-Verlag, Berlin.

[6] Hofert, M. (2011). Efficiently sampling nested Archimedean copulas. *Comput. Stat. Data Anal. 55*(1), 57–70.

[7] Holeňa, M., L. Bajer, and M. Ščavnickỳ (2015). Using copulas in data mining based on the observational calculus. *IEEE T. Knowl. Data Eng. 27*(10), 2851–2864.

[8] Joe, H. (1997). *Multivariate Models and Dependence Concepts*. Chapman & Hall, London.

[9] Kimberling, C. H. (1974). A probabilistic interpretation of complete monotonicity. *Aequationes Math. 10*(2), 152–164.

[10] Matsypura, D., E. Neo, and A. Prokhorov (2016). Estimation of hierarchical archimedean copulas as a shortest path problem. *Econ. Lett. 149*, 131 – 134.

[11] McNeil, A. J. (2008). Sampling nested Archimedean copulas. *J. Stat. Comput. Sim. 78*(6), 567–581.

[12] McNeil, A. J. and J. Nešlehová (2009). Multivariate Archimedean copulas, d-monotone functions and l_1-norm symmetric distributions. *Ann. Stat. 37*(5B), 3059–3097.

[13] Nelsen, R. B. (2006). *An Introduction to Copulas*. Second edition. Springer, New York.

[14] Okhrin, O., Y. Okhrin, and W. Schmid (2013). On the structure and estimation of hierarchical Archimedean copulas. *J. Econometrics 173*(2), 189–204.

[15] Okhrin, O., A. Ristig, and Y. Xu (2016). Copulae in high dimensions: An introduction. Available at https://www.researchgate.net/profile/Yafei_Xu3/publication/309204418_Copulae_in_High_Dimensions_An_Introduction/links/58052cad08aef179365e6dc2.pdf.

[16] Rezapour, M. (2015). On the construction of nested Archimedean copulas for d-monotone generators. *Stat. Probabil. Lett. 101*, 21–32.
[17] Segers, J. and N. Uyttendaele (2014). Nonparametric estimation of the tree structure of a nested Archimedean copula. *Comput. Stat. Data Anal. 72*, 190–204.
[18] Uyttendaele, N. (2016). On the estimation of nested Archimedean copulas: A theoretical and an experimental comparison. Technical report, UCL. Available at https://dial.uclouvain.be/pr/boreal/object/boreal:171500.
[19] Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *J. Am. Stat. Assoc. 58*(301), 236–244.
[20] Zhu, W., C.-W. Wang, and K. S. Tan (2016). Structure and estimation of Lévy subordinated hierarchical Archimedean copulas (LSHAC): Theory and empirical tests. *J. Bank. Financ. 69*, 20–36.