A. Proof of Theorem 1

Consider any tree $T^q \in \mathcal{T}_{T^*}$ and its corresponding set $\mathcal{S}^q$. We find the covariance matrix Σ^q with the same off diagonal elements as Σ^o whose independence structure is given by T^q. Upon obtaining Σ^q, getting the D^q matrix is immediate. To begin with, let us consider the case when $\mathcal{S}^q$ has just one node, i.e, $\mathcal{S}^q$ consists of one of the leaves of T^*.

Proposition 1. *Suppose the covariance matrix Σ^* has conditional independence structure T^* with leaf node a and its neighbor b. Consider a covariance matrix Σ^q defined as follows:*

$$\Sigma^q_{ij} = \begin{cases} \Sigma^*_{ij} - \frac{1}{\Omega^*_{aa}} & \text{if } i = j = a \\ \Sigma^*_{ij} + c^i_1 & 0 < c^i_1 < D^*_{ij} \text{ if } i = j = b \\ \Sigma^*_{ij} & \text{otherwise}, \end{cases}$$

The conditional independence structure T^q of Σ^q is given by the tree obtained by exchanging positions of node a and b in T^*.

Proof. Relabeling if necessary, assume that node n is a leaf node and node $n-1$ is its neighbor in T^*. Define B^1 and B^2 as follows:

$$B^1_{ij} = \begin{cases} c^i_1 & 0 < c^i_1 < D^*_{n-1n-1} \text{ if } i = j = n-1 \\ 0 & \text{otherwise} \end{cases},$$

$$B^2_{ij} = \begin{cases} -\frac{1}{\Omega^*_{nn}} & \text{if } i = j = n \\ 0 & \text{otherwise} \end{cases}.$$

We also define an intermediate matrix $\Sigma^I = \Sigma^* + B^2$. Therefore $\Sigma^q = \Sigma^I + B^1$. The proof of this proposition can be split in the following steps:

(i) We prove that for Σ^I column n is a multiple of column $n-1$ making it a low rank matrix.

(ii) We add B^1 to Σ^I to get Σ^q. In Σ^q column n is a multiple of column $n-1$ at all elements other than $n-1^{st}$. This makes node $n-1$ a leaf node connected to node n as we see in Lemma 1.

(iii) We prove that the independence structure of the rest of the nodes does not change. This is done by proving 2 claims:

  (a) Conditional independence relations do not change when if conditioning is not on node n or node $n-1$.

  (b) Any pair of nodes which were independent conditioned on $n-1$ in Σ^* are independent conditioned on n in Σ^q.

A.1. Proof of Part(i) - Column n of Σ^I is a multiple of column $n-1$:

The precision matrix Ω^* is of the form:

$$\Omega^* = \left[\begin{array}{ccc|c} \Omega^*_{11} & \dots & \Omega^*_{1n-1} & 0 \\ \vdots & \ddots & \vdots & \vdots \\ \Omega^*_{1n-1} & \dots & \Omega^*_{n-1n-1} & \Omega^*_{n-1n} \\ \hline 0 & \dots & \Omega^*_{n-1n} & \Omega^*_{nn} \end{array}\right]. \tag{9}$$

For notational convenience, in what follows, we label the blocks in (9) as Ω^*_x, Ω^*_y and Ω^*_z, so that:

$$\Omega^* = \left[\begin{array}{c|c} \Omega^*_x & \Omega^*_y \\ \hline (\Omega^*_y)^T & \Omega^*_z \end{array}\right].$$

As depicted in (9), block Ω^*_y is a $n-1$ length vector with a non zero only at position $n-1$. The covariance matrix $\Sigma^* = (\Omega^*)^{-1}$ is as follows:

$$\Sigma^* = \left[\begin{array}{ccc|c} \Sigma^*_{11} & \dots & \Sigma^*_{1n-1} & \Sigma^*_{1n} \\ \vdots & \ddots & \vdots & \vdots \\ \Sigma^*_{1n-1} & \dots & \Sigma^*_{n-1n-1} & \Sigma^*_{n-1n} \\ \hline \Sigma^*_{1n} & \dots & \Sigma^*_{n-1n} & \Sigma^*_{nn} \end{array}\right].$$

As with Ω^*, we write it in blocks as:

$$\Sigma^* = \left[\begin{array}{c|c} \Sigma^*_x & \Sigma^*_y \\ \hline (\Sigma^*_y)^T & \Sigma^*_z \end{array}\right]. \tag{10}$$

By the matrix inversion lemma, we have:

$$\Sigma^*_x = (\Omega^*_x)^{-1} + (\Omega^*_x)^{-1}\Omega^*_y[\Omega^*_z - (\Omega^*_y)^T(\Omega^*_x)^{-1}(\Omega^*_y)]^{-1}(\Omega^*_y)^T(\Omega^*_x)^{-1}.$$

To ease notation, we define $c_2 \triangleq [\Omega^*_z - (\Omega^*_y)^T(\Omega^*_x)^{-1}(\Omega^*_y)]^{-1}$. The $(n-1)^{st}$ column of Σ^*_x is given as follows:

$$(\Sigma^*_x)_{:,n-1} = [1 + c_2(\Omega^*_x)^{-1}_{n-1,n-1}(\Omega^*_{n-1n})^2](\Omega^*_x)^{-1}_{:,n-1}. \tag{11}$$

Note that $(\Sigma^*_x)_{n-1,n-1} = \Sigma^*_{n-1n-1}$ and $(\Omega^*_x)_{n-1,n-1} = \Omega^*_{n-1n-1}$.
By the matrix inversion lemma, we also have:

$$\Sigma^*_y = -(\Omega^*_x)^{-1}\Omega^*_y[\Omega^*_z - (\Omega^*_y)^T(\Omega^*_x)^{-1}(\Omega^*_y)]^{-1}.$$

Substituting c_2 for $[\Omega^*_z - (\Omega^*_y)^T(\Omega^*_x)^{-1}(\Omega^*_y)]^{-1}$ and the value of Ω^*_y from equation (9) we get:

$$\Sigma^*_y = -c_2\Omega^*_{n-1n}(\Omega^*_x)^{-1}_{:,n-1}. \tag{12}$$

By Equations (11) and (12) we have:

$$\Sigma^*_y = \frac{-c_2\Omega^*_{n-1n}}{[1 + c_2(\Omega^*_x)^{-1}_{n-1,n-1}(\Omega^*_{n-1n})^2]}(\Sigma^*_x)_{:,n-1}. \tag{13}$$

Hence, the n^{th} column of Σ^* is a multiple of the $(n-1)^{st}$ column except for the n^{th} element. Also, by the matrix inversion lemma $\Sigma^*_{nn} = \Sigma^*_z = c_2$.
Now we look at the intermediate matrix Σ^I which is given as follows:

$$\Sigma^I = \left[\begin{array}{ccc|c} \Sigma^*_{11} & \dots & \Sigma^*_{1n-1} & \Sigma^*_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ \Sigma^*_{1n-1} & \dots & \Sigma^*_{n-1n-1} & \Sigma^*_{n-1n} \\ \hline \Sigma^*_{1n} & \dots & \Sigma^*_{n-1n} & \Sigma^*_{nn} - \frac{1}{\Omega^*_{nn}} \end{array}\right]. \tag{14}$$

Now we prove that Σ^I is a rank deficient matrix and its n^{th} column is a multiple of its $(n-1)^{st}$ column. Specifically, letting $c_3 \triangleq \frac{-c_2\Omega^*_{n-1n}}{[1+c_2(\Omega^*_x)^{-1}_{n-1,n-1}(\Omega^*_{n-1n})^2]}$, we show that $\Sigma^I_{:,n} = c_3\Sigma^I_{:,n-1}$. This is true for the first $(n-1)$ elements by Equation (13).
Basically we need to prove the following:

$$\Sigma^*_{nn} - \frac{1}{\Omega^*_{nn}} = c_3\Sigma^*_{n-1n}. \tag{15}$$

Expanding the LHS in Equation (15), we get

$$\begin{aligned} \Sigma^*_{nn} - \frac{1}{\Omega^*_{nn}} &= \frac{1}{\Omega^*_{nn} - (\Omega^*_{n-1n})^2(\Omega^*_x)^{-1}_{n-1n-1}} - \frac{1}{\Omega^*_{nn}} \\ &= \frac{c_2}{\Omega^*_{nn}}(\Omega^*_{n-1n})^2(\Omega^*_x)^{-1}_{n-1n-1}. \end{aligned} \tag{16}$$

For the RHS of Equation (15), we substitute Σ^*_{n-1n} from Equation (12) and the value of c_3 to get the following:

$$\begin{aligned} c_3\Sigma^*_{n-1n} &= \frac{c_2^2(\Omega^*_{n-1n})^2}{[1 + c_2(\Omega^*_x)^{-1}_{n-1,n-1}(\Omega^*_{n-1n})^2]}(\Omega^*_x)^{-1}_{n-1n-1} \\ &= \frac{c_2(\Omega^*_{n-1n})^2}{[c_2^{-1} + (\Omega^*_x)^{-1}_{n-1,n-1}(\Omega^*_{n-1n})^2]}(\Omega^*_x)^{-1}_{n-1n-1} \\ &= \frac{c_2}{\Omega^*_{nn}}(\Omega^*_{n-1n})^2(\Omega^*_x)^{-1}_{n-1n-1}. \end{aligned} \tag{17}$$

From Equations (16) and (17) we conclude that that $(\Sigma^I)_{:,n} = c_3(\Sigma^I)_{:,n-1}$. Hence, Σ^I is a rank deficient matrix. Also note that the first $n-1$ principal sub matrices of Σ^I have positive determinant by the positive definiteness of Σ^*. Hence, $rank(\Sigma^I) = n-1$.

A.2. Proof of part (ii) - Node $n-1$ is a leaf node connected to node n in the independence structure of Σ^q:

Next we add B^1 to Σ^I to get Σ^q:

$$\Sigma^q = \left[\begin{array}{ccc|c} \Sigma^*_{11} & \dots & \Sigma^*_{1n-1} & \Sigma^*_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ \Sigma^*_{1n-1} & \dots & \Sigma^*_{n-1n-1} + c_1^{n-1} & \Sigma^*_{n-1n} \\ \hline \Sigma^*_{1n} & \dots & \Sigma^*_{n-1n} & \Sigma^*_{nn} - \frac{1}{\Omega^*_{nn}} \end{array}\right],$$

for any $0 < c_1^{n-1} < D^*_{n-1n-1}$. In Σ^q column $n-1$ is not multiple of column n, hence it is a symmetric positive definite matrix making it a valid covariance matrix. Also, column $n-1$ is a multiple at all indices except at index n. In order to prove that node $n-1$ is a leaf node connected to node n, we use Lemma 1.

Lemma 1. *If in any covariance matrix Σ, column $n-1$ is a multiple $\alpha \neq 0$ of column n except at position $n-1$, then in the independence structure of Σ, node $n-1$ is a leaf node connected to node n.*

Proof of Lemma 1: We look at the edges of node $n-1$ given by the $(n-1)^{st}$ column of $\Omega = \Sigma^{-1}$.

$$|\Omega_{n-1i}| = \frac{|det(\Sigma_{-(n-1),-i})|}{det(\Sigma)}$$

For $i \notin n, n-1$, $\Omega_{n-1i} = 0$ as the submatrix $\Sigma_{-(n-1),-i}$ is rank deficient by assumption. Note that $\Omega_{n-1n} \neq 0$, because by contradiction if that was true, Ω would be a block diagonal with node $n-1$ as one block. This would imply that Σ would be a block diagonal with node $n-1$ as one block, which cannot be the case as $\Sigma_{n-1n} = \alpha\Sigma_{nn} \neq 0$. Hence node $n-1$ is a leaf node connected to node n. □

By Lemma 1, node $n-1$ is a leaf node connected to node n in T^q.

A.3. Proof of part (iii) - Structure of the remaining tree does not change:

In order to prove this part, we need the following lemma:

Lemma 2. *For any random vector $Y = [Y_1, Y_2, \dots, Y_n]$, $Y \sim \mathcal{N}(0, \Sigma)$, Y_i is independent of Y_j conditioned on Y_k if and only if*

$$\Sigma_{ij} = \frac{\Sigma_{ik}\Sigma_{jk}}{\Sigma_{kk}}.$$

Proof of Lemma 2: The probability distribution of Y_{-k} conditioned on Y_k is given as follows:

$$Y_{-k} \mid Y_k \sim \mathcal{N}(\Sigma_{-k,k}\Sigma_{kk}^{-1}Y_k, \Sigma_{-k,-k} - \frac{\Sigma_{k,-k}\Sigma_{-k,k}}{\Sigma_{kk}}).$$

For Y_i to be independent of Y_j conditioned on Y_k, the i,j component of the conditional covariance matrix must be zero, giving

$$\Sigma_{ij} = \frac{\Sigma_{ik}\Sigma_{jk}}{\Sigma_{kk}}. \quad \square$$

Proof of part (iiia) - Conditional independence relations, when conditioning is not on n or $n-1$, don't change:
This is a direct consequence of Lemma 2 as $\Sigma^q_{kk} = \Sigma^*_{kk}$ for $k \neq n, n-1$.

Proof of part (iiib) - Any pair of nodes which were independent conditioned on $n-1$ in Σ^* are independent conditioned on n in Σ^q:

Suppose node i and node j were independent conditioned on node $n-1$ in Σ^* and $i, j \neq n$. Then by Lemma 2 we have:

$$\Sigma^*_{ij} = \frac{\Sigma^*_{n-1i}\Sigma^*_{n-1j}}{\Sigma^*_{n-1n-1}}.$$

From Equation(10), note that $\Sigma^*_{n-1i} = (\Sigma^*_x)_{n-1i}$ and $\Sigma^*_{n-1j} = (\Sigma^*_x)_{n-1j}$, also $\Sigma^*_{ni} = (\Sigma^*_y)_i$ and $\Sigma^*_{nj} = (\Sigma^*_y)_j$. So, by Equation (13), we have:

$$\Sigma^*_{ij} = \frac{\Sigma^*_{ni}\Sigma^*_{nj}}{c_3\Sigma^*_{n-1n}}.$$

Since the off diagonal terms of Σ^* and Σ^q are equal, we have:

$$\Sigma^q_{ij} = \frac{\Sigma^q_{ni}\Sigma^q_{nj}}{c_3\Sigma^q_{n-1n}}.$$

By Equation (15) we can substitute the denominator to obtain:

$$\Sigma^q_{ij} = \frac{\Sigma^q_{ni}\Sigma^q_{nj}}{\Sigma^q_{nn}}.$$

Therefore, by Lemma 2, in the graphical structure for Σ^q, i and j are independent conditioned on n. □

Proving parts (i), (ii) and (iii) proves Proposition 1, that the conditional independence structure of Σ^q is given by the tree T^q. For a leaf node a and its neighbor b in T^*, the decomposition $\Sigma^o = \Sigma^q + D^q$ which results in the exchange to nodes a and b is as follows:

$$\Sigma^q_{ij} = \begin{cases} \Sigma^*_{ij} - \frac{1}{\Omega^*_{aa}} & \text{if i = j = a} \\ \Sigma^*_{ij} + c^i_1 & 0 < c^i_1 < D^*_{ij} \text{ if } i = j = b \\ \Sigma^*_{ij} & \text{otherwise,} \end{cases}$$

$$D^q_{ii} = \begin{cases} D^*_{ii} + \frac{1}{\Omega^*_{aa}} & \text{if } i = a \\ D^*_{ii} - c^i_1 & \text{if } i = b \\ D^*_{ii} & \text{otherwise,} \end{cases}$$

□

Thus far, we have only considered the case when $\mathcal{S}_q$ has just one node. This analysis directly extends to the case when $\mathcal{S}_q$ has more than one nodes. The Σ^q and D^q matrices in that case are as follows:

$$\Sigma^q_{ij} = \begin{cases} \Sigma^*_{ij} - \frac{1}{\Omega^*_{ij}} & \text{if } i = j \in \mathcal{S}^q \\ \Sigma^*_{ij} + c^i_1 & \text{if } i = j \in Neighbor(\mathcal{S}^q) \\ \Sigma^*_{ij} & \text{otherwise,} \end{cases}$$

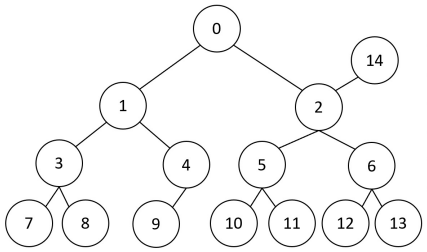
$$D^q_{ii} = \begin{cases} D^*_{ii} + \frac{1}{\Omega^*_{ii}} & \text{if } i \in \mathcal{S}^q \\ D^*_{ii} - c^i_1 & \text{if } i \in Neighbor(\mathcal{S}^q) \\ D^*_{ii} & \text{otherwise,} \end{cases}$$

where $Neighbor(\mathcal{S}^q)$ is the set of neighbor nodes of all the nodes in $\mathcal{S}^q$. Also, c^i_1 is chosen such that $0 < c^i_1 < D^*_{ii}$. This completes the proof of Theorem 1. □

B. Proof of Theorem 2

We prove this theorem by proving that the off diagonal terms of covariance matrix are enough to determine the structure of the underlying tree up to the equivalence set $\mathcal{T}_{T^*}$. The main building block of this proof and of the algorithm presented in Section 5 is to categorize any set of 4 nodes as a star shape or a non-star shape. Moreover, if it is a non star star shape we further divide the set of 4 nodes in half forming 2 pairs of nodes.

Definition 4.
- *Four nodes $\{i_1, i_2, i_3, i_4\}$ form a **non-star shape** if there exists a node i_k in the tree T^*[2] such that exactly two nodes among the four lie in the same connected component of $T^* \setminus i_k$.*
- *If $\{i_1, i_2, i_3, i_4\}$ does not form a non-star shape, we say they form a **star shape**.*



Node Set	Classification	i_k, pairing (If Applicable)
{0, 1, 3, 7}	Non-Star	1, {{0,1}, {3,7}}
{7, 8, 9, 10}	Non-Star	1, {{7,8}, {9,10}}
{14, 2, 10, 11}	Non-Star	2, {{14,2}, {10,11}}
{7, 9, 1, 6}	Star	N.A.
{1, 14, 10, 6}	Star	N.A.

Figure 6. Examples of classification of 4 nodes as star shape or non-star shape.

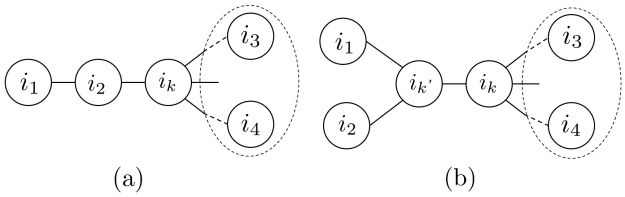


Figure 7. Conditional independence for non-star shape

It is easy to see that in the event that a set of 4 nodes forms a non star, there exists a grouping such that the 2 nodes in the same connected component form the first pair and the other 2 nodes form the second pair. Examples of star shape and non-star shape are presented in Figure 6. This categorization is done using only the off-diagonal elements of the covariance matrix, hence this property remains invariant to diagonal perturbations, that is, every set of 4 nodes falls in the same category in any tree obtained from the decomposition of $\Sigma^o = \Sigma' + D'$ as $\Sigma'_{ij} = \Sigma^*_{ij}\ \forall i \neq j$.
The proof of this theorem is split in 3 parts:

(i) Prove that it is possible to categorize any set of 4 nodes as star shape or non-star shape using only off diagonal elements of the covariance matrix.

(ii) Prove that this categorization of 4 nodes completely defines all the possible partitions of the original tree in 2 connected components such that the connected components have at least 2 node.

(iii) Prove that these partitions of a tree into connected components completely define the tree structure up to the equivalence set $\mathcal{T}_{T^*}$.

B.1. Proof of Part (i) - Categorization of 4 nodes as star/non-star shape:

We first state the conditions using only off-diagonal elements for a set of 4 nodes to be categorized as non-star shape. Assume that a set of 4 nodes $\{i_1, i_2, i_3, i_4\}$ satisfy the definition of a non-star shape such that nodes i_1 and i_2 form one pair and i_3 and i_4 form the second pair. This is true if and only if:

$$\begin{aligned} &\frac{\Sigma^*_{i_1 i_3}}{\Sigma^*_{i_1 i_4}} = \frac{\Sigma^*_{i_2 i_3}}{\Sigma^*_{i_2 i_4}}, \\ &\frac{\Sigma^*_{i_2 i_1}}{\Sigma^*_{i_3 i_1}} \neq \frac{\Sigma^*_{i_2 i_4}}{\Sigma^*_{i_3 i_4}} \text{ and} \\ &\frac{\Sigma^*_{i_2 i_1}}{\Sigma^*_{i_4 i_1}} \neq \frac{\Sigma^*_{i_2 i_3}}{\Sigma^*_{i_3 i_4}}. \end{aligned} \tag{18}$$

The first equality and the second inequality imply the last inequality. When nodes $\{i_1, i_2, i_3, i_4\}$ form a non star shape, they either satisfy a conditional independence structure shown in Figure 7(a) or 7(b) for some nodes i_k and $i_{k'}$.
For Figure 7(a), the following conditional independence relations hold:

$$i_1 \perp i_3, i_4 | i_2, \tag{19}$$

[2]Note that nothing prevents i_k to be one of the four nodes.

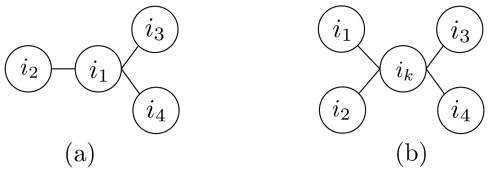


Figure 8. Conditional independence for star shape.

$$i_3 \not\perp i_4 | i_2. \tag{20}$$

Using Lemma 2, we get the following conditions for the conditional independence relation in Equations (19) and (20):

$$\Sigma^*_{i_2 i_2} = \frac{\Sigma^*_{i_1 i_2}\Sigma^*_{i_3 i_2}}{\Sigma^*_{i_1 i_3}} = \frac{\Sigma^*_{i_1 i_2}\Sigma^*_{i_4 i_2}}{\Sigma^*_{i_1 i_4}} \neq \frac{\Sigma^*_{i_3 i_2}\Sigma^*_{i_4 i_2}}{\Sigma^*_{i_3 i_4}}. \tag{21}$$

Using Equation (21) we get the relations in Equation (18).
For Figure 7(b), the following conditional independence relations hold:

$$i_1 \perp i_3, i_4 | i_{k'}, \tag{22}$$

$$i_2 \perp i_3, i_4 | i_{k'}, \tag{23}$$

$$i_3 \not\perp i_4 | i_{k'}. \tag{24}$$

Using Lemma 2, we get the following conditions for the conditional independence relation in Equations (22), (23) and (24):

$$\Sigma^*_{i_{k'} i_{k'}} = \frac{\Sigma^*_{i_1 i_{k'}}\Sigma^*_{i_3 i_{k'}}}{\Sigma^*_{i_1 i_3}} = \frac{\Sigma^*_{i_1 i_{k'}}\Sigma^*_{i_4 i_{k'}}}{\Sigma^*_{i_1 i_4}} = \frac{\Sigma^*_{i_2 i_{k'}}\Sigma^*_{i_3 i_{k'}}}{\Sigma^*_{i_2 i_3}} = \frac{\Sigma^*_{i_2 i_{k'}}\Sigma^*_{i_4 i_{k'}}}{\Sigma^*_{i_2 i_4}} \neq \frac{\Sigma^*_{i_3 i_{k'}}\Sigma^*_{i_4 i_{k'}}}{\Sigma^*_{i_3 i_4}}. \tag{25}$$

Using Equation (25), we get the conditions in Equation (18). Note that for both the cases in Figure 7, the Equation (18) remains the same if i_1 and i_2 exchange positions.
Next, we state the conditions using only off-diagonal elements for a set of 4 nodes to be categorized as a star shape. Assume that a set of 4 nodes $\{i_1, i_2, i_3, i_4\}$ satisfy the definition of a star shape. This is true if and only if:

$$\begin{aligned} \frac{\Sigma^*_{i_1 i_3}}{\Sigma^*_{i_1 i_4}} &= \frac{\Sigma^*_{i_2 i_3}}{\Sigma^*_{i_2 i_4}}, \\ \frac{\Sigma^*_{i_2 i_1}}{\Sigma^*_{i_3 i_1}} &= \frac{\Sigma^*_{i_2 i_4}}{\Sigma^*_{i_3 i_4}} \text{ and} \\ \frac{\Sigma^*_{i_2 i_1}}{\Sigma^*_{i_4 i_1}} &= \frac{\Sigma^*_{i_2 i_3}}{\Sigma^*_{i_3 i_4}}. \end{aligned} \tag{26}$$

First 2 equalities imply the third equality. Any set of 4 nodes $\{i_1, i_2, i_3, i_4\}$ can form a star structure only if their conditional independence relation is given by Figure 8(a) or 8(b) for some node i_k. For Figure 8(a), the conditional independence relations are given as:

$$i_2 \perp i_3, i_4 | i_1, \tag{27}$$

$$i_3 \perp i_4 | i_1. \tag{28}$$

Using Lemma 2, we get the following for these conditional independence relations in Equations (27) and (28):

$$\Sigma^*_{i_1 i_1} = \frac{\Sigma^*_{i_1 i_2}\Sigma^*_{i_1 i_3}}{\Sigma^*_{i_2 i_3}} = \frac{\Sigma^*_{i_1 i_2}\Sigma^*_{i_1 i_4}}{\Sigma^*_{i_2 i_4}} = \frac{\Sigma^*_{i_1 i_4}\Sigma^*_{i_1 i_3}}{\Sigma^*_{i_4 i_3}}. \tag{29}$$

Equation (29) implies Equation (26).
For Figure 8(b), the conditional independence relations are given as:

$$i_1 \perp i_2, i_3, i_4 | i_k, \tag{30}$$

$$i_2 \perp i_3, i_4 | i_k, \tag{31}$$

$$i_3 \perp i_4 | i_k. \tag{32}$$

Using Lemma 2, we get the following for the conditional independence relations in Equations (30), (31) and (32):

$$\Sigma^*_{i_k i_k} = \frac{\Sigma^*_{i_1 i_k}\Sigma^*_{i_2 i_k}}{\Sigma^*_{i_1 i_2}} = \frac{\Sigma^*_{i_1 i_k}\Sigma^*_{i_3 i_k}}{\Sigma^*_{i_1 i_3}} = \frac{\Sigma^*_{i_1 i_k}\Sigma^*_{i_4 i_k}}{\Sigma^*_{i_1 i_4}} = \frac{\Sigma^*_{i_2 i_k}\Sigma^*_{i_3 i_k}}{\Sigma^*_{i_2 i_3}} = \frac{\Sigma^*_{i_2 i_k}\Sigma^*_{i_4 i_k}}{\Sigma^*_{i_2 i_4}} = \frac{\Sigma^*_{i_3 i_k}\Sigma^*_{i_4 i_k}}{\Sigma^*_{i_3 i_4}}. \tag{33}$$

Equation (33) implies Equation (26).
Hence using only the off diagonal terms, checking the conditions in Equations (18) and (26), any set of 4 nodes can be classified as a star shape or non-star shape. □

B.2. Proof of Part (ii) - Partitioning of the tree in 2 connected components:

We prove this by presenting an explicit algorithm to obtain a specific partition of the original tree T^*, which would also be a valid partition of T', using the categorization of any set of 4 nodes as a star shape or non-star shape. This procedure can be performed with different initializations to obtain all the possible partitions.
Let $\mathcal{A}$ denote the set of all the nodes in T^*.

Definition 5. *A **subtree** $\mathcal{B}$ of a tree T^* is a set of nodes such that $\mathcal{B}$ and $\mathcal{A} \setminus \mathcal{B}$ form a connected component in T^*. The pair of subtrees $\mathcal{B}$ and $\mathcal{A} \setminus \mathcal{B}$ are called complementary subtrees.*

For any set of 4 nodes $\{i_1, i_2, i_3, i_4\}$ that form a non-star shape such that nodes i_1 and i_2 form a pair, we obtain the smallest subtree containing i_1 and i_2 by Algorithm 1. Basically, we fix i_1, i_2 and i_3 and scan through all the remaining nodes to form a set of 4 nodes and check if it forms a star or non-star shape. If this set of 4 nodes forms a star shape or forms a non-star shape such that the scanned node pairs with i_1 or i_2, we put it in group 1, otherwise, we put it in group 2. Once we are done scanning through all the nodes, group 1 gives the smallest subtree and group 2 gives its complementary subtree.

Algorithm 1 Partition all the nodes in complementary subtrees.

Input - Observed Covariance Matrix (Σ^o), Set of 4 nodes($\{i_1, i_2, i_3, i_4\}$)
Output - The smallest subtree containing i_1 and i_2($group1$) and the complementary subtree ($group2$).

procedure SMALLESTSUBTREE($\Sigma^o, \{i_1, i_2, i_3, i_4\}$)
  $n\_rows \leftarrow size(\Sigma^o, 1)$
  $index \leftarrow \{i_1, i_2, i_3, 0\}$
  for $j = 1$ to n_rows **do**
    if j in $group1$ or $group2$ **then**
      continue
    $index[4] = j$
    $status, pair1, pair2 \leftarrow$ ISSTARSHAPE($index, \Sigma^o$)
    if $status$ **then** ▷ If $\{i_1, i_2, i_3, j\}$ forms a star shape, add j to $group1$.
      $group1.append(j)$
    else
      if j pairs with $index[3]$ **then** ▷ If j pairs with i_3, add j to $group2$.
        $group2.append(j)$
      else
        $group1.append(j)$ ▷ Otherwise add j to $group1$.
  return $group1, group2$

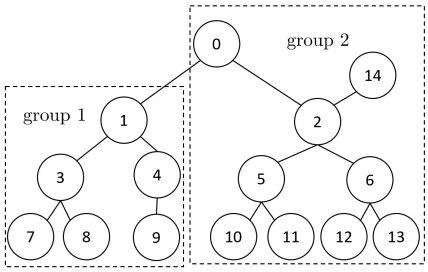


Figure 9. Suppose $i_1 = 7$, $i_2 = 9$ and $i_3 = 5$. If j is in group 2, $\{i_1, i_2, i_3, j\}$ is categorized as a non star and j pairs with i_3. If j is in group 1, $\{i_1, i_2, i_3, j\}$ is either categorized as a star or it is categorized as a non star and j pairs with i_1 or i_2.

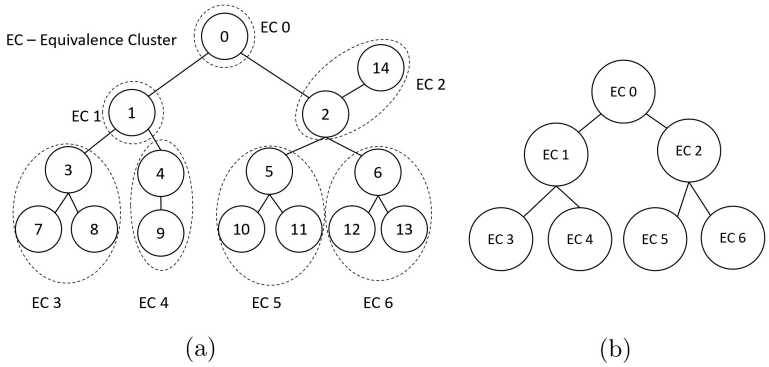


Figure 10. (a) Equivalence clusters for the given tree. (b) The cluster tree with equivalence clusters as vertices.

PROOF OF CORRECTNESS OF ALGORITHM 1

Consider the tree T^*. We denote the smallest subtree containing nodes i_1 and i_2 by $\mathcal{B}$. Let $i_{k'}$ denote the node in $\mathcal{B}$ that has an edge with the connected component formed by $\mathcal{A} \setminus \mathcal{B}$. Let i_k be the node in $\mathcal{A} \setminus \mathcal{B}$ that has an edge with a node in $\mathcal{B}$. In this case i_k is a node such that nodes i_1 and i_2 lie in the same connected component of $T^* \setminus i_k$. By the definition of non-star shape, i_3 cannot be in $\mathcal{B}$. Also, a node j can be in $\mathcal{A} \setminus \mathcal{B}$ if and only if nodes $\{i_1, i_2, i_3, j\}$ are non star and j pairs with i_3 as nodes i_1 and i_2 still lie in the same connected component of $T^* \setminus i_k$. This is illustrated in Figure 9.
Using different i_1 and i_2, we get all the possible partitions of the tree T^*.

B.3. Proof of Part (iii) - Recovering the tree up to unidentifiability using tree partitions

Before going to the proof of this part, we define the terms equivalence cluster, cluster tree, cluster subtrees, complementary cluster subtrees and the root of a cluster subtree as follows:

Definition 6. *A set containing an internal node and all the leaf nodes connected to it forms an **equivalence cluster**. We say that there is an edge between two equivalence clusters if there is an edge between any node in one equivalence cluster and any node in the other equivalence cluster. An equivalence cluster which has an edge with at most one more equivalence cluster is called a leaf equivalence cluster.*

Definition 7. *A tree with equivalence clusters as vertices and edges between equivalence clusters as the edges is called a **cluster tree**.*

Example of equivalence clusters and a cluster tree are presented in Figure 10. The cluster tree completely defines the set $\mathcal{T}_{T^*}$.

Definition 8. *A **cluster subtree** is a set where the equivalence clusters are plugged in for the corresponding nodes in a subtree. Complementary cluster subtrees are the subtrees obtained when this is done for a pair of complementary subtrees.*

Definition 9. *The **root of a cluster subtree** is the equivalence cluster that has an edge with the complementary cluster subtree.*

To prove this theorem we show that the partitions obtained in part (ii) completely define the cluster tree. We call the subtrees obtained from part (ii) input subtrees. Note that each input subtree has at least 2 nodes. We prove this in 2 steps:

(i) The input subtrees define the equivalence clusters.

(ii) The input subtrees define the edges between the equivalence clusters.

ALGORITHM TO FIND EQUIVALENCE CLUSTERS

The algorithm to find the equivalence clusters takes all the input subtrees and performs the following steps:

1. Initialize the set of discovered equivalence clusters as an empty set.
2. Identify one input subtree which does not have a subset of nodes forming another input subtree. This input subtree forms an equivalence cluster. Append it to the list of equivalence clusters.
3. Construct trimmed subtrees by removing the equivalence cluster from the input subtrees.
4. Repeat steps 2 and 3 with trimmed subtrees as input subtrees.

PROOF OF CORRECTNESS:

We prove the correctness of this algorithm by induction on the number of equivalence clusters.
Base Case ($k = 1$):
When there is 1 equivalence cluster, there is 1 input subtree and it is the equivalence cluster.
Inductive Step:
Assume the algorithm works for a tree with k or less equivalence clusters. We prove that the algorithm works for a tree with $k + 1$ equivalence clusters.
Relabeling if necessary, assume that $k + 1$ is a leaf equivalence cluster. Hence it forms a subtree and no subset of the equivalence cluster can form a subset of another input subtree (as the smallest input subtree which contains at least 2 of these nodes is the whole equivalence cluster). Thus in Step 2, $k + 1$ is recognized as an equivalence cluster.
By trimming in Step 3, we remove the $k + 1^{st}$ equivalence cluster from all the subtrees. Hence, we are left with a tree with k equivalence clusters. By inductive assumption, the algorithm can find these k equivalence clusters. Therefore, the algorithm finds all the $k + 1$ equivalence clusters.

ALGORITHM TO FIND THE EDGES BETWEEN EQUIVALENCE CLUSTERS

For this part we identify the root of every cluster subtree as follows:
An equivalence cluster is the root of a cluster subtree if and only if, upon its removal, the remaining elements can be written as a union of smaller cluster subtrees which are a subset of the original cluster subtree.
To prove this claim, assume that we remove an equivalence cluster other than the root. In that case the root must have an edge with the complementary cluster subtree and hence it cannot be obtained by a union of smaller cluster subtrees which are a subset of the original cluster subtrees.
The algorithm to find the edges between equivalence clusters performs the following steps:

1. Initialize the set of edges as a null set and the set of unexplored complementary cluster subtrees as the set of all the complementary cluster subtrees.
2. Select a pair of complementary cluster subtrees from the set of unexplored complementary cluster subtrees.
3. Find the root nodes of both the cluster subtrees and append an edge between the two roots in the set of edges.
4. Trim the currently selected cluster subtrees from all the cluster subtrees in the unexplored set for which the currently explored cluster subtrees are a subset(this also deletes the currently selected cluster subtrees from the unexplored set). Repeat Steps 2, 3 and 4 with the trimmed cluster subtrees till the unexplored set is empty.

PROOF OF CORRECTNESS:

We prove the correctness of this algorithm by induction on the number of equivalence clusters.
Base Case ($k = 2$): In this case there are 2 cluster subtrees which are complementary cluster subtrees. Both of them have 1

equivalence cluster which is also the root. Hence the algorithm finds the edge between the two cluster subtrees.
Inductive Step: Suppose the algorithm works for a tree with k or less equivalence clusters. We prove that the algorithm works for a tree with $k+1$ equivalence clusters.
Relabeling if necessary, assume that $k+1$ is a leaf equivalence cluster. Hence there exists a pair of complementary cluster subtrees where one cluster subtree contains the $k+1$ equivalence cluster and the other cluster contains the first k equivalence cluster. Hence the edge of the $(k+1)^{st}$ equivalence cluster is added to the list of edges. Once this edge is recognized, the $(k+1)^{st}$ equivalence cluster is trimmed and the algorithm correctly finds the edges of the remaining cluster tree by the inductive assumption.

Hence the input subtrees completely define the equivalence clusters and the edges between them. This completes the proof of theorem 2. □

C. Proof of Theorem 4

To prove this claim, we consider the decomposition of $\Sigma^o = \Sigma' + D'$ such that the conditional independence structure T' for Σ' has leaf node b and its neighbor node a. We show that $\Omega'_{bb} < |\Omega'_{ab}|$, that is, the leaf node b in T' violates the constraint. Hence, any decomposition of Σ^o which results in an exchange of a leaf node with its neighbor is infeasible. Therefore, the problem becomes identifiable.

Relabeling if necessary, assume that node n is a leaf node connected to node $n-1$ in T^*. Recall that the decomposition of $\Sigma^o = \Sigma' + D'$ from Proposition 1 to obtain a tree structure T' in which node $n-1$ is a leaf node connected to node n is given by:

$$\Sigma'_{ij} = \begin{cases} \Sigma^*_{ij} - \frac{1}{\Omega^*_{ij}} & \text{if } i = j = n \\ \Sigma^*_{ij} + c & 0 < c < D^*_{n-1n-1} \text{ if } i = j = n-1 \\ \Sigma^*_{ij} & \text{otherwise.} \end{cases}$$

We derive the expression of $\Omega' = (\Sigma')^{-1}$. We denote B_1 and B_2 as follows:

$$B_1 = \begin{cases} c & 0 < c < D^*_{n-1n-1} \text{ if } i = j = n-1 \\ 0 & \text{otherwise} \end{cases},$$
$$B_2 = \begin{cases} -\frac{1}{\Omega^*_{nn}} & \text{if } i = j = n \\ 0 & \text{otherwise} \end{cases}.$$

This gives us $\Sigma' = \Sigma^* + B_1 + B_2$. Hence Σ' is Σ^* plus a rank 2 matrix. To calculate its inverse, we first evaluate:

$$\begin{aligned} (\Sigma^* + B_1)^{-1} &= \Omega^* - \frac{1}{1 + tr(\Omega^* B_1)} \Omega^* B_1 \Omega^* \\ &= \Omega^* - \frac{c\Omega^*_{:,n-1}\Omega^*_{n-1,:}}{1 + c\Omega^*_{n-1n-1}}. \end{aligned} \tag{34}$$

We next evaluate Ω' as follows:

$$\Omega' = (\Sigma^* + B_1 + B_2)^{-1} = (\Sigma^* + B_1)^{-1} - \frac{1}{1 + tr((\Sigma^* + B_1)^{-1} B_2)} (\Sigma^* + B_1)^{-1} B_2 (\Sigma^* + B_1)^{-1}.$$

This expression can be simplified by substituting the value of $(\Sigma^* + B_1)^{-1}$ from Equation (34) to arrive at:

$$\Omega' = \Omega^* + \frac{(1 + c\Omega^*_{n-1n-1})}{c(\Omega^*_{n-1n})^2} \Omega^*_{:,n}\Omega^*_{n,:} - \frac{1}{\Omega^*_{n-1n}} (\Omega^*_{:,n-1}\Omega^*_{n,:} + \Omega^*_{:,n}\Omega^*_{n-1,:}). \tag{35}$$

Now we look at the terms in positions $(n-1, n-1)$ and $(n-1, n)$ of Ω'.

$$\begin{aligned}\Omega'_{n-1n-1} &= \Omega^*_{n-1n-1} + \frac{(1 + c\Omega^*_{n-1n-1})}{c} - 2\Omega^*_{n-1n-1} \\ &= \frac{1}{c}. \\ \Omega'_{n-1n} &= \Omega^*_{n-1n} + \frac{(1 + c\Omega^*_{n-1n-1})}{c\Omega^*_{n-1n}}\Omega^*_{nn} - \Omega^*_{n-1n} - \frac{\Omega^*_{nn}\Omega^*_{n-1n-1}}{\Omega^*_{n-1n}} \\ &= \frac{\Omega^*_{nn}}{c\Omega^*_{n-1n}}.\end{aligned}$$

By the original assumption we have $\Omega^*_{nn} > |\Omega^*_{n-1n}|$, hence $\Omega'_{n-1n-1} < |\Omega'_{n-1n}|$. Therefore the leaf node $n-1$ in T' violates the additional constraint and hence this decomposition of Σ^o is infeasible. Extending the argument, any decomposition of Σ^o which results in a tree T' in which leaf node of T^* exchanges position with its neighbor is infeasible. Hence T^* and T' have the same structure. □

D. Proof of Theorem 6

To prove this theorem, we consider Σ' such that the conditional independence structure has b as the leaf node and a as its neighbor. Rest of the struture is the same as T^*. We find a lower bound on the minimum eigenvalue of Σ', λ'_{min}. If this lower bound is greater than λ_{min}, this implies that there exists a feasible decomposition which has conditional independence structure different from T^*.

In order to lower bound the minimum eigenvalue of Σ', we upper bound the maximum eigenvalue of Ω'. We do this using a corollary of Gerschgorin's Theorem. We use the result that the maximum eigenvalue of Ω' is upper bounded by the maximum of the sum of absolute values of all the row entries:

$$\frac{1}{\lambda'_{min}} \leq \max_i \Big(\sum_{j=1}^{n} |\Omega'_{ij}| \Big). \tag{36}$$

From the expression of Ω' stated in Equation (35) (by relabeling the nodes n and $n-1$ as nodes a and b respectively), we have:

$$\sum_{j=1}^{n} |\Omega'_{ij}| = \begin{cases} \frac{1}{c}\Big(\frac{(\Omega^*_{aa})^2}{(\Omega^*_{ab})^2} + \frac{\Omega^*_{aa}}{\Omega^*_{ab}}\Big) + \frac{\Omega^*_{aa}(\Omega^*_{aa}\Omega^*_{bb} - (\Omega^*_{ab})^2)}{(\Omega^*_{ab})^2} + \sum_{\substack{j=1 \\ j\neq a,b}}^{n} \frac{\Omega^*_{aa}|\Omega^*_{qj}|}{|\Omega^*_{ab}|} & \text{if } i = a, \\ \frac{1}{c}\Big(1 + \frac{\Omega^*_{aa}}{\Omega^*_{ab}}\Big) & \text{if } i = b. \\ \Big(\sum_{\substack{j=1 \\ j\neq a,b}}^{n} |\Omega^*_{ij}| + \frac{\Omega^*_{aa}|\Omega^*_{qi}|}{|\Omega^*_{ab}|}\Big) & \text{otherwise.} \end{cases}$$

Using the definitions in Equation 6, we can rewrite the upper bound in Equation (36) as follows:

$$\frac{1}{\lambda'_{min}} \leq \max\big(\frac{e^{ab}}{c}, \frac{f^{ab}}{c} + g^{ab}, h^{ab}\big).$$

Rewriting this as:

$$\frac{1}{\lambda'_{min}} \leq \begin{cases} \frac{e^{ab}}{c} & \text{if } c \leq \frac{e^{ab} - f^{ab}}{g^{ab}} \\ \frac{f^{ab}}{c} + g^{ab} & \text{if } \frac{e^{ab} - f^{ab}}{g^{ab}} < c \leq \frac{f^{ab}}{h^{ab} - g^{ab}} \\ h^{ab} & otherwise. \end{cases}$$

First, let us concentrate on the first case. For unidentifiability, we need:

$$c \geq e^{ab}\lambda_{min}.$$

To remain in the first case, we need $c \leq \frac{e^{ab} - f^{ab}}{g^{ab}}$. Therefore, if $\lambda_{min} \leq \frac{(e^{ab} - f^{ab})}{e^{ab}g^{ab}}$ and $D^*_{bb} \geq e^{ab}\lambda_{min}$, there would exist a feasible value of c which allows node a and b to switch positions.

Next we look at the second case. If $\lambda_{min} < \frac{1}{g^{ab}}$, for unidentifiability, we need:

$$c \geq \frac{f^{ab}}{1/\lambda_{min} - g^{ab}}.$$

To remain in the second case, we need $c \leq \frac{f^{ab}}{h^{ab}-g^{ab}}$. Therefore, if $\lambda_{min} < \frac{1}{h^{ab}}$ and $D^*_{bb} \geq \frac{f^{ab}}{1/\lambda_{min}-g^{ab}}$, there would exist a feasible value of c which allows node a and b to switch positions. If $\lambda_{min} > \frac{1}{g^{ab}}$, nothing can be said about unidentifiability. To enter the third case, we need $\lambda_{min} > \frac{1}{h^{ab}}$ which would again imply that nothing could be said about identifiability.

E. Algorithms

E.1. Pseudo-code

We give the pseudo-code for all the functions introduced in Section 5. Note that we have adopted the convention that the indexing starts from 1.

Algorithm 2 Determine whether any set of 4 nodes is star shape or non-star shape.

procedure IsStarShape($index = \{i_1, i_2, i_3, i_4\}, \Sigma^o$)
  $sub\_sigma \leftarrow \Sigma^o[index; index]$ ▷ submatrix of Σ^o with only the input nodes.
  $count\_pairs \leftarrow 0$ ▷ Count the number of column pairs which satisfy ratio condition.
  for $column_1 = 1$ to 3 **do**
    for $column_2 = column_1 + 1$ to 4 **do**
      $is\_ratio\_initialized \leftarrow False$
      for $z = 1$ to 4 **do**
        if $z == column_1$ or $z == column_2$ **then** ▷ Skip diagonal elements.
          continue
        if Not($is\_ratio\_initialized$) **then**
          $ratio \leftarrow sub\_sigma[z, column_2]/sub\_sigma[z, column_1]$ ▷ Calculate ratio of elements.
          $is\_ratio\_initialized \leftarrow True$
        else
          if $ratio == sub\_sigma[z, column_2]/sub\_sigma[z, column_1]$ **then**
            $count\_pairs \leftarrow count\_pairs + 1$ ▷ Counts pairs with equal ratio.
            if $count\_pairs == 1$ **then**
              $pair1 = [index[column_1], index[column_2]]$
  if $count\_pairs == 2$ **then**
    $pair2 \leftarrow index \setminus pair1$ ▷ Nodes not in $pair1$ form $pair2$.
    return $False, pair1, pair2$ ▷ non-star shape.
  else
    return $True$, [ ], [ ] ▷ Star Shape.

E.2. Proof of correctness

E.2.1. Proof for Algorithm 2: IsStarShape

The correctness of algorithm 2 is already proven in section B.1.

E.2.2. Proof for Algorithm 3: PartitionNodes

The procedure in Algorithm 3 is initialized by finding a combination of four nodes which form a non-star shape. To achieve that, we fix two nodes and scan through all the possible pairs of remaining nodes. In order to prove that this is enough we look at the different configurations of two fixed nodes and argue the existence of two other nodes which can make the 4 node set a non-star shape (if such a shape exists in the tree).

Let the fixed nodes be i_1 and i_2. We now study all the different cases which can arise:

Algorithm 3 Partition all the nodes in 2 subtrees.

procedure PARTITIONNODES(Σ^o)
  $n\_rows \leftarrow size(\Sigma^o, 1)$
  $index \leftarrow$ [1,2,0,0]
  $is\_star \leftarrow True$
  for $i_3 = 3$ to $n\_rows - 1$ **do**
    for $i_4 = i_3 + 1$ to n_rows **do**
      $index[3] \leftarrow i_3$
      $index[4] \leftarrow i_4$
      $is\_star, pair1, pair2 \leftarrow$ ISSTARSHAPE($index, \Sigma^o$)
      if NOT(is_star) **then** ▷ We found a non-star shape.
        break from both loops
  if is_star **then** ▷ We did not find any non-star shape.
    return is_star,[ ], [ ]
  else ▷ We found one non star. We use it to partition the nodes.
    $group1, group2 \leftarrow$ SMALLESTSUBTREE($\Sigma^o, index$)
  return $is\_star, group1, group2$

Algorithm 4 Get a node in subtree $\mathcal{B}$ from the equivalence cluster closest to the external node $i_{outside}$.

procedure GETCLOSESTEQUIVALENCECLUSTER($\mathcal{B}, i_{outside}, \Sigma^o$)
  $n\_node \leftarrow len(\mathcal{B})$
  $index \leftarrow [i_{outside}, 0, 0, 0]$
  $i_{close} \leftarrow \mathcal{B}[1]$ ▷ Initial estimate of a node from closest EC.
  for $j_{candidate}$ = 2 to n_node **do** ▷ Sweep through $j_{candidate}$ to find a node from the closest EC.
    $index[2] \leftarrow i_{close}$
    $i_{candidate} \leftarrow \mathcal{B}[j_{candidate}]$
    $index[3] \leftarrow i_{candidate}$
    for $j = 1$ to n_node **do** ▷ Sweep through j until we find a non-star shape.
      if $\mathcal{B}[j]$ in $index$ **then**
        continue
      $index[4] = \mathcal{B}[j]$
      $is\_star, pair1, pair2 \leftarrow$ ISSTARSHAPE($index, \Sigma^o$)
      if NOT(is_star) **then**
        break
    if $i_{candidate}$ pairs with $i_{outside}$ **then** ▷ i_{close} ruled out. $i_{candidate}$ is the new estimate
      $i_{close} \leftarrow i_{candidate}$
  $equivalence\_cluster \leftarrow [i_{close}]$ ▷ i_{close} is in the EC closest to $i_{outside}$.
  for $i_{equivalent} \in \mathcal{B} \setminus i_{close}$ **do** ▷ Find other nodes of the closest EC.
    $all\_star\_shapes \leftarrow True$
    for $j \in \mathcal{B} \setminus \{i_{close}, i_{equivalent}\}$ **do**
      $is\_star, \sim, \sim \leftarrow$ ISSTARSHAPE($\{i_{outside}, i_{close}, i_{equivalent}, j\}, \Sigma^o$)
      if NOT(is_star) **then**
        $all\_star\_shapes \leftarrow False$
        break
    if all_star_shapes **then** ▷ If $i_{equivalent}$ always form a star shape, it is in the EC of i_{close}.
      Add $i_{equivalent}$ to $equivalence\_cluster$
  return $equivalence\_cluster$

Algorithm 5 Splits $\mathcal{B} \setminus EC_{close}$ into the set of largest subtrees.

procedure SPLITROOTEDTREE($i_{outside}, EC_{close}, \mathcal{B}, \Sigma^o$)
  $subtrees$ =[ ] ▷ $subtrees$ is a list of lists where each list contains the nodes of one subtree.
  $\mathcal{B} \leftarrow \mathcal{B} \setminus EC_{close}$ ▷ Remove the EC of i_{root} from $\mathcal{B}$.
  $i_{close} \leftarrow EC_{close}[1]$
  Create first subtree $\mathcal{B}_1$ with $\mathcal{B}[1]$ ▷ Initialize $\mathcal{B}_1$ with any node of $\mathcal{B}$.
  for $j \in \mathcal{B}$ **do**
    $is\_star \leftarrow True$
    for $\mathcal{B}_i \in subtrees$ **do**
      $index \leftarrow [i_{outside}, i_{close}, j, \mathcal{B}_i[1]]$ ▷ Check if new node j forms non star with subtree $\mathcal{B}_i$.
      $is\_star, pair1, pair2 \leftarrow$ ISSTARSHAPE($index, \Sigma^o$)
      if NOT(is_star) **then**
        break
    if NOT(is_star) **then**
      Add j to $\mathcal{B}_i$ ▷ Add new node j to the last subtree $\mathcal{B}_i$ before the break.
    else
      Create new subtree with j ▷ Create new subtree with only j.
  return $subtrees$

Algorithm 6 Recursive function finds equivalence clusters and edges between them.

procedure LEARNEDGES($i_{outside}, \mathcal{B}, learned\_edges, equivalence\_clusters, \Sigma^o$)
  $n\_nodes \leftarrow len(\mathcal{B})$
  if $n\_nodes == 2$ **then** ▷ $\mathcal{B}$ is an equivalence cluster.
    Add $\mathcal{B}$ to $equivalence\_cluster$
    Add $(EC(i_{outside}), EC(\mathcal{B}[1]))$ to $learned\_edges$
    return
  $EC_{close} \leftarrow$ GETCLOSESTEQUIVALENCECLUSTER($\mathcal{B}, i_{outside}, \Sigma^o$) ▷ Get the closest EC.
  Add EC_{close} to $equivalence\_cluster$
  Add $(EC(i_{outside}), EC_{close})$ to $learned\_edges$ ▷ Add an edge between EC containing $i_{outside}$ and EC_{close}
  $subtrees \leftarrow$ SPLITROOTEDTREE($i_{outside}, EC_{close}, \mathcal{B}, \Sigma^o$) ▷ Get subtrees of $\mathcal{B} \setminus \{EC_{close}\}$.
  for $\mathcal{B}_j \in subtrees$ **do**
    LEARNEDGES($i_{close}, \mathcal{B}_j, learned\_edges, equivalence\_clusters, \Sigma^o$) ▷ Recursive call for all the subtrees.

Algorithm 7 Full Algorithm.

procedure LEARNTREESTRUCTURE(Σ^o)
  $learned\_edges \leftarrow$ [ ]
  $equivalence\_custers \leftarrow$ [[ ]]
  $\mathcal{A} \leftarrow \{1 \dots n\}$
  $is\_star, \mathcal{B}, \mathcal{B}' \leftarrow$ PARTITIONNODES(Σ^o)
  if is_star **then** $equivalence\_custers \leftarrow \mathcal{A}$
    return $equivalence\_custers, learned\_edges$
  else
    $i_{outside_{\mathcal{B}}} \leftarrow$ GETCLOSESTEQUIVALENCECLUSTER($\mathcal{B}'[1], \mathcal{B}, \Sigma^o$)
    $i_{outside_{\mathcal{B}'}} \leftarrow$ GETCLOSESTEQUIVALENCECLUSTER($\mathcal{B}[1], \mathcal{B}', \Sigma^o$)
    LEARNEDGES($i_{outside_{\mathcal{B}}}, \mathcal{B}, learned\_edges, equivalence\_clusters, \Sigma^o$)
    LEARNEDGES($i_{outside_{\mathcal{B}'}}, \mathcal{B}', learned\_edges, equivalence\_clusters, \Sigma^o$)
  return $equivalence\_clusters, learned\_edges$

- If i_1 and i_2 are leaves with different neighbors, a combination of the two leaves with their neighbors forms a non star.
- If i_1 and i_2 are leaves with a common neighbor, and there exists another leaf with a different neighbor, i_1 and i_2 and the other leaf neighbor pair form a non-star shape. If there does not exist another leaf with a different neighbor, then the tree has one equivalence cluster, and no non-star shape exists.
- If i_1 and i_2 are internal nodes, combining them with one node from the connected component of $T^* \setminus i_1$ that contains i_2 and another node from a different connected component of $T^* \setminus i_1$ gives a non star shape.
- If one of i_1 and i_2 is an internal node and the other one is a leaf node and the internal node is not a neighbor of the leaf node, combining them with the neighbor of the leaf node and another leaf with a different neighbor gives a non-star shape.
- If the internal node is the neighbor of the leaf node, combining them with another pair of leaf and neighbor gives a non star structure.

When we obtain the initial set of 4 nodes which form a non star structure, we also obtain the pairing of the 4 nodes. Performing the procedure of Algorithm 3 splits the tree into the smallest subtree that contains pair 1 and the remaining subtree.

E.2.3. PROOF FOR ALGORITHM 4: GETCLOSESTEQUIVALENCECLUSTER

Let i_{root} be the node in $\mathcal{B}$ which has an edge with a node in $\mathcal{A} \setminus \mathcal{B}$ and i_{close} be a node from the equivalence cluster containing i_{root}.

Lemma 3. *The set $\{i_{outside}, i_{close}, i_1, i_2\}$ forms a non star shape if and only if i_1 and i_2 lie in one connected component of $\mathcal{B} \setminus i_{root}$.*

Corollary 2. *When $\{i_{outside}, i_{close}, i_1, i_2\}$ forms a non star shape i_1, i_2 form one pair and $i_{outside}, i_{close}$ form the second pair.*

Proof. The set $\{i_{outside}, i_{close}, i_1, i_2\}$ forms a non star if $\exists\, i_k$ such that exactly 2 of these nodes lie in the same connected component of $\mathcal{B} \setminus i_k$.
Proof of If:
Setting $i_k = i_{root}$ gives us the non star shape for this set. Moreover, i_1, i_2 form one pair and $i_{outside}, i_{close}$ form the other pair.
Proof of Only if:
We now prove that if i_1 and i_2 are not in the same subtree of $\mathcal{B} \setminus i_{root}$, then $\{i_{outside}, i_{close}, i_1, i_2\}$ does not form a non-star shape, *i.e.* it is impossible to find a node i_k such that exactly two nodes of $\{i_{outside}, i_{close}, i_1, i_2\}$ are in the same subtree of $T^* \setminus i_k$. We look at the possible i_k we could choose:

- If $i_k \notin \mathcal{B}$, $\{i_{close}, i_1, i_2\}$ are in the same subtree of $T^* \setminus i_k$.
- $i_k = i_{root}$, then $i_{outside}$, i_1 and i_2 are in different subtrees of of $T^* \setminus i_k$.
- If i_k is in one of the connected component of $\mathcal{B} \setminus i_{root}$, then at least one of the two nodes i_1 or i_2 is not in the same component. Therefore, either $\{i_{outside}, i_{close}, i_1\}$, $\{i_{outside}, i_{close}, i_2\}$ or $\{i_{outside}, i_{close}, i_1, i_2\}$ are together in the same subtree of $T^* \setminus i_k$.

Hence there is no i_k such that exactly 2 of $\{i_{outside}, i_{close}, i_1, i_2\}$ lie in the same connected component of $T^* \setminus i_k$. Therefore $\{i_{outside}, i_{close}, i_1, i_2\}$ forms a star shape. □

Any node $i_{equivalent}$ is in the equivalence cluster containing i_{root}, if and only if any set of 4 nodes $\{i_{equivalent}, j, i_{outside}, i_{close}\}$ $\forall\ j \in \mathcal{B}$ forms a star shape. By Lemma 3, this set of 4 nodes forms a star shape if and only if $i_{equivalent}$ and j do not lie in the same connected component of $\mathcal{B} \setminus i_{root}$. Thus $i_{equivalent}$ is either i_{root} or a leaf node connected to i_{root}. Hence $i_{equivalent}$ lies in the equivalence cluster containing i_{root}.

E.2.4. PROOF OF ALGORITHM 5: SPLITROOTEDTREE

Given an external node $i_{outside}$, the equivalence cluster EC_{close} containing i_{root} and a node i_{close} from EC_{close}, by Lemma 3, $\{i_{outside}, i_{close}, i_1, i_2\}$ forms a non-star shape if and only if i_1 and i_2 are in the same connected component of $\mathcal{B} \setminus i_{root}$. This is used to find all the subtrees in $\mathcal{B} \setminus EC_{close}$.

E.2.5. PROOF OF ALGORITHM 6: LEARNEDGES

We show that LEARNEDGES$(i_{outside}, \mathcal{B}, learned_edges, equivalence_clusters, \Sigma^o)$ correctly learns the equivalence clusters in $\mathcal{B}$ and the edges between these equivalence clusters as well as the edge between the equivalence cluster containing $i_{outside}$ and the equivalence cluster in $\mathcal{B}$ closest to $i_{outside}$. We do this by induction on the number of equivalence clusters.
Base case: $\mathcal{B}$ contains 1 equivalence cluster
Note that the function GETCLOSESTEQUIVALENCECLUSTER needs at least 3 nodes in $\mathcal{B}$. The base case can be split in 2 cases:
Case 1: If $\mathcal{B}$ has 2 nodes, it has to contain a leaf node and its neighbor, hence it forms one equivalence cluster which is identified and an edge is added between the EC containing $i_{outside}$ and the EC in $\mathcal{B}$.
Case 2: If $\mathcal{B}$ has more than 2 nodes, the equivalence cluster is correctly identified by GETCLOSESTEQUIVALENCECLUSTER. An edge is added between the EC containing $i_{outside}$ and the EC in $\mathcal{B}$.
Inductive step: Let the function identify all the equivalence clusters and edges when $\mathcal{B}$ has less than n equivalence clusters. Now suppose $\mathcal{B}$ has n equivalence clusters. By the correctness of GETCLOSESTEQUIVALENCECLUSTER, it correctly identifies the equivalence cluster EC_{close} in $\mathcal{B}$ with the node that has an edge with $EC(i_{outside})$ and adds this edge. By the correctness of SPLITROOTEDTREE, it correctly identifies all the subtrees in $\mathcal{B} \setminus EC_{close}$. All these subtrees have less than n equivalence clusters. By the inductive assumption, the function correctly learns all the edges between EC_{close} and the closest equivalence clusters in these subtrees as well as all the edges within these subtrees.

E.2.6. PROOF OF ALGORITHM 7: LEARNTREESTRUCTURE

By the correctness of PARTITIONNODES, we successfully partition the whole tree in two subtrees. By the correctness of Algorithm GETCLOSESTEQUIVALENCECLUSTER, we find the equivalence clusters in these subtrees which connect to the other subtree. By the correctness of LEARNEDGES, we accurately discover the equivalence clusters in these 2 subtrees, the edges between these equivalence clusters as well as the edge between the equivalence clusters of the 2 subtrees. This gives us all the equivalence clusters and the edges between them.

E.3. Running Time Analysis

ISSTARSHAPE is $\mathcal{O}(1)$ operation.
PARTITIONNODES is $\mathcal{O}(n^2)$ as in the worst case when the tree is star structured, it needs to search through all the pairs of nodes.
GETCLOSESTEQUIVALENCECLUSTER is $\mathcal{O}(n^2)$ as it checks all the nodes once for being better than the current estimate of connecting node. Checking this at each step involves scanning through all the nodes till a non star structure is discovered which in the worst case can take $\mathcal{O}(n)$ time. It further finds all the other nodes from the equivalence cluster. To do that, it scans through all the nodes and checks if that node can form a non star. Checking if it can form a non star is $\mathcal{O}(n)$. Hence the complexity is $\mathcal{O}(n^2)$.
SPLITROOTEDTREE is $\mathcal{O}(n^2)$ as the outer for loop scans through all the nodes in the input subtree and the inner loop scans through one node from all the output subtrees. Both of these are $\mathcal{O}(n)$ is worst case. Hence the complexity is $\mathcal{O}(n^2)$.

LEARNEDGES is $\mathcal{O}(n^3)$ as it calls GETCLOSESTEQUIVALENCECLUSTER and SPLITROOTEDTREE at most $n-1$ times.
LEARNTREESTRUCTURE is $\mathcal{O}(n^3)$ as it calls LEARNEDGES twice.