8-2019

Learning Topologies of Acyclic Networks with Tree Structures

Firoozeh Sepehr
University of Tennessee

To the Graduate Council:

I am submitting herewith a dissertation written by Firoozeh Sepehr entitled "Learning Topologies of Acyclic Networks with Tree Structures." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Computer Science.

Donatello Materassi, Major Professor

We have read this dissertation and recommend its acceptance:

Michael Langston, Seddik Djouadi, Hamparsum Bozdogan

Accepted for the Council:

Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Learning Topologies of Acyclic Networks with Tree Structures

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Firoozeh Sepehr

August 2019

Acknowledgments

I would like to give a special thanks to my advisor, Dr Donatello Materassi, for his advice and support throughout all my experience as a PhD student. I would not be where I am today without him inspiring me at every step of this incredible and challenging journey. Also, thank you to my committee, Dr Djouadi, Dr Langston and Dr Bozdogan, for agreeing to review my work and for their guidance and feedback on my work. I would also like to thank my family, friends, and colleagues who believed in me and supported me throughout all these years.

My PhD research has been partially funded by the National Science Foundation: NSF (CNS CAREER #1553504) and I am so grateful for their support of my work.

Abstract

Network topology identification is known as the process of revealing the interconnections of a network where each node is representative of an atomic entity in a complex system. This procedure is an important topic in the study of dynamic networks since it has broad applications spanning different scientific fields. Furthermore, the study of tree structured networks is deemed significant since a large amount of scientific work is devoted to them and the techniques targeting trees can often be further extended to study more general structures. This dissertation considers the problem of learning the unknown structure of a network when the underlying topology is a directed tree, namely, it does not contain any cycles.

The first result of this dissertation is an algorithm that consistently learns a tree structure when only a subset of the nodes is observed, given that the unobserved nodes satisfy certain degree conditions. This method makes use of an additive metric and statistics of the observed data only up to the second order. As it is shown, an additive metric can always be defined for networks with special dynamics, for example when the dynamics is linear. However, in the case of generic networks, additive metrics cannot always be defined. Thus, we derive a second result that solves the same problem, but requires the statistics of the observed data up to the third order, as well as stronger degree conditions for the unobserved nodes. Moreover, for both cases, it is shown that the same degree conditions are also necessary for a consistent reconstruction, achieving the fundamental limitations. The third result of this dissertation provides a technique to approximate a complex network via a simpler one when the assumption of linearity is exploited. The goal of this approximation is to highlight the most significant connections which could potentially reveal more information about the network. In order to show the reliability of this method, we consider high frequency financial data and show how well the businesses are clustered together according to their sector.

Table of Contents

List of Tables

List of Figures

Nomenclature

- := denotes a definition
- $\{x, y\}$ unordered pair
- (x, y) ordered pair
- y node
- y_h hidden node
- y_r root node
- y_c child node
- y_p parent node
- n number of nodes
- n_r number of root nodes
- G undirected graph
- $\vec{G}$ directed graph
- $\bar{G}$ partially directed graph
- $\vec{T}$ rooted tree graph
- $\vec{P}$ polytree graph
- $\vec{F}$ polyforest graph

- N set of nodes (or vertices)
- V set of visible nodes
- L set of hidden nodes
- E set of undirected edges (or arcs)
- $\vec{E}$ set of directed edges (or arcs)
- $|A|$ number of elements in the set A
- S subset of nodes
- C hidden cluster
- $N(C)$ neighbors of the hidden cluster C
- $\deg_G(y)$ degree of the node y in graph G
- $\deg^+_{\vec{G}}(y)$ outdegree of the node y in graph $\vec{G}$
- $\deg^-_{\vec{G}}(y)$ indegree of the node y in graph $\vec{G}$
- $\mathrm{pa}_{\vec{G}}(y)$ parents of the node y in graph $\vec{G}$
- $\mathrm{ch}_{\vec{G}}(y)$ children of the node y in graph $\vec{G}$
- $\mathrm{an}_{\vec{G}}(y)$ ancestors of the node y in graph $\vec{G}$
- $\mathrm{de}_{\vec{G}}(y)$ descendants of the node y in graph $\vec{G}$
- u wide-sense stationary stochastic process
- $\mathbf{I}$ identity matrix
- $H(z)$ transfer matrix
- $\mathbb{Z}(\cdot)$ Zeta-transform of a signal
- A^* conjugate transpose of the matrix A

- $\mathbb{E}[\cdot]$ expected value
- $R_{xy}(\tau) := \mathbb{E}[x(t)y^T(t+\tau)]$ cross-covariance function
- $R_x(\tau) := R_{xx}(\tau)$ autocovariance
- $\Phi_x(z) := \Phi_{xx}(z)$ power spectral density of signal x
- $\Phi_{xy}(z) := \mathbb{Z}(R_{xy}(\tau))(z)$ cross-power spectral density of signals x and y
- $x \perp\!\!\!\perp y$: $\Phi_{xy}(z) = 0$
- $\mathcal{I}(y_i, y_k, y_j)$ independence statement - y_i and y_j are conditionally independent given y_k
- $\neg\mathcal{I}(y_i, y_k, y_j)$ negation of $\mathcal{I}(y_i, y_k, y_j)$
- $\mathbb{P}$ probability

Chapter 1

Introduction

Networks of dynamic systems have become a widespread modeling tool with applications spanning fields as diverse as physics [5, 6], biology [7], chemistry [8], medicine [9], neuropsychology [10], ecology [11, 12], economics [13, 14], engineering [15] and social networks [16]. For example, in [10] a learning methodology is developed that establishes the interconnections between different brain regions. This is an important procedure since cognitive tasks recruit multiple regions of the brain and therefore understanding how these regions are affecting each other will help characterize neural basis of cognitive processes. As another example, the authors of [16] apply an algorithm to create the causal diagram of the trending topics discussed by popular Twitter handles. This causal diagram is then used to identify the trend setters, namely, the users that have influenced other users the most by starting a topic.

The first step of studying a network of dynamic systems is typically to identify how its internal processes (or nodes) are connected to each other. This problem might be tackled under different scenarios. A first scenario considers the situations when excitations are used to probe the network and receive its response in order to identify the network structure; these methods are commonly known as active reconstruction in the literature [17]. A second scenario considers the situations when the inputs of the network are measurable but not adjustable; these scenarios are often known as non-invasive reconstruction [17, 18]. A third and more challenging scenario is when the inputs of the network are not measurable at all and the only observable part of the system is its outputs; these techniques are known as blind reconstruction [19, 20]. In the latter scenario, the measurements of the outputs are not the system response to known inputs and data

are acquired while the system is operating and forced by potentially unknown excitations. Since blind reconstruction methods identify a network only from observations of the outputs, they have practical applications even in large scale networks fulfilling critical or uninterruptible functions, such as power grids [21] or logistic systems [22], and also in situations where it is impractical or too expensive to inject known probing signals into the system, such as gene or financial networks [23, 24]. Furthermore, the applications of these techniques span the field of medicine such as repeated drug testing [25], automatically assisted anesthesia [26], and deep brain stimulation for Parkinson's disease [3, 27].

The ultimate objective of this dissertation is to propose novel algorithms to learn the structure of a network using only the observations of the network outputs (blind reconstruction technique). To achieve this goal, we develop three main algorithms under different assumptions. Parts of these results are already published in [28], [2] and [3] while [1] and [4] are currently under review for publication. In the following sections, we review some of the relevant work in the literature and then present an overview of the contributions of this dissertation.

1.1 Techniques to Learn Network Structures

Several algorithms have been developed with the goal of learning the structure of a network from observational data. These algorithms are mostly derived in the area of graphical models to describe conditional independence relations among random variables [29, 30]. Only more recently techniques have been developed in the domain of stochastic processes to describe input/output relations among dynamic systems [17, 19, 20, 31, 32, 33]. Graphical models of random variables and networks of dynamic systems have inherently different underlying semantics. However, it is shown that many of the techniques developed for learning the structure of a network can be consistently applied to both fields [34]. This is also the objective of this dissertation: develop algorithms to learn the network structure of random variables of a graphical model and also the network structure of stochastic processes of a dynamic system [3].

In the area of graphical models, different approaches have been proposed to learn the structure of a network [35, 36, 37]. In [36], these methods are categorized in three different approaches: (i) constraint-based structure learning, where the network is viewed as a representation of

dependencies; (ii) score-based structure learning, where the network is viewed as a statistical model and the structure is learned via a model selection approach; (iii) model averaging, where an ensemble of possible structures are generated. Here, we are interested in studying approaches that fall into the first category. Among the constraint-based structure learning tools, one of the most versatile approaches is the SGS (Spirtes, Glymour, and Scheines) algorithm developed to infer a Bayesian network of random variables from data [37]. This algorithm provides a consistent reconstruction of the topology of a Bayesian network described by a Directed Acyclic Graph (DAG). However, it cannot, in the general case, determine the orientation of all the links in the graph. Moreover, one fundamental drawback of the SGS algorithm is that it relies on several searches of subsets of the graph nodes resulting in exponential time complexity with respect to the number of nodes. Variations of this algorithm such as the PC (Peter, Clark) algorithm are developed to exploit the conditional independence relations to reduce the computational complexity. However, its worse case scenario still runs in exponential time with respect to the highest degree of the nodes, again, making it not suitable to deal with large networks.

In contrast, a different set of approaches make use of a priori information about the structure by deriving reconstruction algorithms with better scaling and sample complexity properties. A widely used algorithm to approximate a discrete probability distribution with tree factorization was developed by Chow and Liu in [38]. If a distribution has a tree factorization, it means that each factor is the conditional probability distribution given at most one other variable (namely, a product of first or second order distributions). This strategy has been successfully employed in biology for the study of gene regulatory networks to approximate a complex structure with a tree topology [23]. Also, in economics, this method has been applied to identify a tree network for the analysis of a stock portfolio [14]. Other techniques such as phylogenetic reconstruction approach have been developed that utilize a metric defined over pairs of nodes of a binary tree with applications in biology [39].

Similarly, in the area of dynamic systems, there are algorithms capable of reconstructing quite large classes of networks [20, 33, 34, 40, 41, 42, 43, 44, 45]. For example, the authors of [20], propose an approach to consistently reconstruct the structure of an unknown dynamic network using spectral factorization methods for stable, minimum-phase Linear Time Invariant (LTI) systems. The reconstructed network, in this case, is unique given that the system is strictly

causal. Furthermore, the authors also propose a method to deal with non-minimum phase systems with strictly causal dynamics, though, in this case the solution is not necessarily unique. Aside from the strong assumption of strict causality, these approaches rely on the spectral factorizations which do not scale very well with the number of nodes.

In [34], the authors show that a modification of the PC algorithm can be applied to reconstruct linear networks of dynamic systems given that the structure is a DAG. Similar to the PC algorithm, this modified version is guaranteed to be consistent but it also suffers from the same limitations in orienting the edges in the network. Other approaches are developed using Granger causality to learn the structure of a network of time series data [19, 46]. The result in [46] achieves this goal, with applications in econometrics, when the noise processes are assumed to be white while the method in [19] further utilizes Wiener filtering with no assumption on the color of the noise processes.

The authors of [42] develop a framework for reconstruction of networks of stochastic processes using the Compressed Sensing Theory (CST) with applications to propagation of diseases or rumors. Computational cost is often a limiting factor for the practical implementation of techniques aiming at reconstructing generic networks such as the one in [42]. Thus, in order to keep the computational cost of the reconstruction algorithm at tractable levels, the authors limit themselves to strictly causal binary stochastic processes [42]. Other results formulate the reconstruction problem looking specifically for a sparse solution via compressed sensing tools such as [43] and [44]. However, the main drawback of these techniques lies in the fact that it is often difficult to find guarantees for the correct reconstruction when applying CST.

In [40], the authors introduce a metric that is a function of the coherence of the pairs of signals and use this metric to develop a technique that reconstructs the skeleton of a dynamic network with a rooted tree structure similar to the approach developed in [38] for graphical models of random variables. They also provide guarantees that this method learns the correct interconnections between the graph nodes, resulting in a correct reconstruction of the structure. Moreover, it is shown that this algorithm could be utilized to provide an approximation of a complex network with a single rooted tree. Such an approximation is shown to be optimal, since it minimizes the mutual information between pairs of nodes in the original network, and also consistent, since it is

shown that if the original network is a rooted tree, then the output of the algorithm converges to the actual network structure in the limit of infinite data.

The algorithms discussed so far have been developed under the assumption that all of the variables in the network are observed. However, as it is often the case, there might be some variables that are not measurable, though the dynamics of the network is affected by their presence. In the next section, we provide a review of the existing methods dealing with these hidden processes and discuss their strengths and weaknesses.

1.2 Learning Techniques for Networks with Unmeasurable (Hidden) Nodes

A fundamental and interesting challenge for learning the structure of networks both in the area of graphical models of random variables and the dynamic systems of stochastic processes occurs when only part of the nodes of the network are observable. This is a relevant issue in many different fields when dealing with practical applications of learning a network structure. For example, in biological networks some of the nodes might not be measurable while they could potentially play a relevant role in the network dynamics [47]. As an additional example, attacks in cyber-security applications are often described by modeling the intruders as hidden (or latent) nodes injecting malicious information into other nodes or stealing information from them [48].

In the study of dynamic systems or control theory, hidden nodes are typically associated only with unmeasured state components [49, 50]. There are numerous techniques to reconstruct a network considering the hidden states as the unmeasured variables with a pronounced attention towards computational efficiency. A common tool to handle hidden states in networks of stochastic processes is the Dynamic Structure Function (DSF) framework developed in [33, 51, 52]. This approach treats all state components that are not measured as hidden nodes. Thus, DSF is merely an input/output network representation of the observable state components connected by transfer functions where the unobserved variables are marginalized and therefore do not appear in the final output of this method.

However, in the graphical models area hidden nodes have a different meaning since they represent unmeasurable parts of the network where new information is being introduced and are not just unmeasured states [53]. In the scope of this dissertation, even when dealing with dynamic systems, we consider hidden nodes as points where new information is added to the network. Thus, in this respect, our approaches bear more similarities with graphical model tools than with control theoretic tools. Considering this definition of hidden nodes, despite the differences in semantics of dynamic systems and graphical models as mentioned in previous section, in some cases similar learning methodologies can be applied to both domains, such as the results of this dissertation [3] or the results in [54].

The first algorithms which could detect the presence of hidden nodes took advantage of specific statistical tests called spectral quartet tests. Spectral quartet tests are effective only when learning tree structures [35, 55, 56] or bipartite Bayesian networks of binary variables [57]. A departure from spectral quartet tests was an algorithm developed in [58] which could learn a binary tree just by the observation of the leaves. The authors of [58] propose a different approach making use of a metric that is additive along the paths of a rooted tree. A generalization of the technique in [58] was later achieved with the Recursive Grouping Algorithm (RGA). RGA also leverages an additive metric defined over pairs of nodes to reconstruct the structure of a generic rooted tree network from observational data [53]. RGA learns the exact structure of the tree so long as the degree of each hidden node is greater than or equal to three and therefore is not limited to the assumption that all visible nodes are the leaves of the tree. Similar methods that take advantage of an additive metric are developed in the case of discrete distributions such as Bernoulli which can be extended to Gaussian models as well [59]. It is noteworthy to mention that the methods developed for learning the tree structured networks have polynomial computational complexity. Another algorithm called Learning Pairwise Cluster Comparison (LPCC) proposes a solution for learning the networks of discrete variables with no prior assumption on the distribution [60, 61]. However, this method makes the strong assumption that no observable node can be an ancestor of any hidden node, limiting the number of networks that can be recovered by LPCC.

In the case of generic distributions, though, finding an additive metric is extremely hard or such a metric might not even exist in general. Many algorithms have been developed to solve this problem for generic networks including cycles in the presence of unmeasured variables [62, 63,

64]. One prevalent method is the use of ancestral graphs to describe the independence relations among the observed variables given that the true network is a DAG [64]. The main advantage of ancestral graphs is that they utilize only the measured variables and successfully encode all of their conditional independence relations via m-separation statements, which generalize the well-known criterion for d-separation [35, 63]. Furthermore, complete algorithms have been devised to obtain ancestral graphs from observational data [63]. However, recovering the actual structure of the original DAG considering the presence of the hidden variables is a task that ancestral graphs somehow circumvent. This means that the exact location and number of the hidden nodes would still be unknown after the recovery technique has been applied. A similar method, known as ancestral polytree, while providing efficiency in the inference process, is developed for cases when the ancestral graph has a polytree (directed tree with potentially multiple roots) structure [65]. Yet, there exist polytree networks such that their ancestral polytree graphs do not have a polytree structure. Therefore, these cases cannot be handled by the ancestral polytree method, limiting the number of different classes of networks that the algorithm can learn.

A different and recent methodology to learn the location and connectivity of hidden nodes in a network of dynamic systems with a polytree structure is described in [66]. This method is based on a discrepancy measure which is a function of the mutual information between pairs of nodes and it relies on the estimation of high order statistics requiring, in general, large quantities of data. The algorithm is applicable, again, only when each link of the network is strictly causal. Considering strictly causal dynamics is a very limiting assumption due to several reasons: (i) transfer functions with direct feedthroughs such as proportional gains are very common; (ii) many discretization methods for continuous systems lead to necessarily non-strictly causal operators; (iii) when delays are smaller than the sampling rate, correlations might appear as instantaneous in the collected data: in these cases strictly causal relations would not be appropriate to describe the relations among the node processes [2, 3].

Observe that many of the different techniques discussed in this section study tree structured networks. In the next section, we discuss why these structures are important and why we are also interested in studying this type of networks.

1.3 The Importance of Tree Structured Networks

The significance of studying acyclic structures, such as trees, is supported in the literature by the large amount of scientific articles devoted to these models [21, 38, 40, 56, 66, 67, 68, 69, 70]. Even though acyclic structures are a relatively limited class of networks, there exist well established tools to extend techniques developed for acyclic structures to cyclic networks such as junction tree approaches [70, 71]. Thus, these results constitute, potentially, a first step towards the development of techniques applicable to more general networks. As an additional example, belief propagation was developed only for trees at first [67], but it was further generalized for loopy networks afterwards [72]. Furthermore, acyclic structures are extensively studied because they can be used to approximate more complex networks. While there could be methods to consistently reconstruct more general classes of networks, these approaches tend to have higher computational and sample complexity. Thus, given a complex system, it might be sometimes preferable to approximate it with a simpler structure. Some examples of these procedures are shown in [14, 23, 40] where a whole gene network and the underlying connectivity of hundreds of financial time series are successfully approximated with a rooted tree. These examples signify the importance of developing fast and efficient algorithms for learning networks with tree structures [1, 2, 3, 4].

Although rooted tree topologies can be satisfactory models in applications where propagations arise from a single source [21], they do not necessarily perform well in applications where information is fused from multiple sources. Examples of these scenarios are in complex power systems where it is possible to generate power in different points inside the distribution grid [73], or social networks where multiple nodes can be the source of misinformation [74, 75]. Polyforest structures (collections of directed trees with potentially multiple roots) have the capacity to model processes that are not necessarily correlated and in fact represent a wider range of network classes modeling arbitrarily high order statistics [68]. For these reasons, in this dissertation we propose novel methodologies to learn the structure of polyforest networks [1, 2, 3, 4].

In the next section, we consider a problem where all the nodes are observable and instead the interest lies in finding an approximation of the network using polytrees in order to capture the strongest connections.

1.4 Approximation using Simpler Structures

As mentioned in previous sections, there are several techniques allowing the exact reconstruction of networks from blind observations considering specific assumptions. While these methods can learn potentially complicated networks and provide guarantees of a consistent learning, they rely on a large quantity of observations for an accurate estimation of conditional independence relations, power spectral factors or evaluation of many coefficients in several multivariate linear regressions [1]. On the other hand, rooted trees have proven to be good topology approximators in several application domains where the actual underlying network is definitely more complex [14, 23]. Since these approximators have tree structures, the computational cost of the approximation method is drastically lower compared to the exact methods.

Another important advantage of using simpler structures as approximators is that when multiple models satisfactorily explain the data, the simpler network is often optimal with respect to some measure, for example a distance defined over pairs of nodes in the network. Thus, this approximation usually tends to have fewer number of edges following a form of Occam's razor principle. This simpler structure is in some cases preferred over recovering the actual structure because a network with fewer connections can potentially highlight the most significant connections between the nodes of the system. Indeed, in these cases a network with fewer edges and supposedly less explanatory power could be paradoxically more informative in terms of how a system operates compared to a network that achieves a marginally better explanatory power by introducing a large number of weak connections.

In the next section we discuss the contributions of this dissertation and also explain how tree structured networks are leveraged for this study.

1.5 Contributions of this Dissertation

As a first contribution, this dissertation considers the problem of learning the unknown structure of a linear dynamic network when the underlying topology is given by a polyforest and some nodes are not observable. No assumption is made about the strict causality of the dynamic operators and only statistics up to the second order are used. It is shown that the proposed methodology is robust

with respect to the presence of unmeasured nodes. In other words, the derived algorithm detects the exact number and location of the latent nodes if they satisfy specific degree conditions in the actual network graph. It is also shown that the required degree conditions are necessary for a consistent reconstruction. Thus, the proposed learning algorithm achieves the fundamental limitations in learning the structure of a polyforest network of linear dynamic systems in the presence of latent nodes [2, 3]. This technique tackles the problem in an efficient way since the computational complexity of the derived algorithm is proven to be polynomial in the number of observed nodes. This method splits a polytree into its rooted trees and then leverages RGA to recover all the hidden nodes that have degree greater than or equal to three in each rooted tree. Furthermore, this method is capable of detecting some additional hidden nodes with degree equal to two which RGA cannot detect. We introduce an algorithm similar to the one introduced in [68] to find the orientation of some of the links either by extracting available features from the data or exploiting some a priori knowledge. Furthermore, it is shown that the proposed method developed for learning polyforest structures in the case of linear networks of dynamic systems can be applied to the case of Gaussian random variables for which we can define a distance metric with the property of being additive along the paths of the rooted trees of the polyforest.

The second contribution of this work is to propose a novel methodology towards the recovery of networks with signals generated by general distributions. Indeed, we provide an algorithm to learn causal diagrams with polyforest structures making no assumption on the underlying probability distribution or linearity of the dynamics of the network processes. These polyforest structures can represent factorizations involving conditional distributions of arbitrarily high order. The proposed technique, remarkably, uses only the statistics of the observable nodes up to the third order. It is shown that a causal diagram with polyforest structure can be exactly recovered if and only if each hidden node in the original diagram satisfies specific degree conditions. These degree conditions are stronger compared to the degree conditions for learning the structure of a network with linear dynamics since the assumption of linearity is relaxed. Moreover, if the degree conditions are not satisfied, it is shown that there exists another polyforest with fewer number of hidden nodes which entails the same independence relations among the observed variables. Therefore, this algorithm, similar to the first proposed algorithm, achieves the fundamental limitations of solving the problem

of learning the structure of a polyforest network with the presence of hidden nodes, given the aforementioned a priori assumptions [4].

The third contribution of this work is to leverage simpler networks with polytree structures as an approximating class of potentially complex networks assuming that all the nodes are observable. This technique is focused on obtaining theoretical guarantees only for cases where the original structure is also a polytree. A basic requirement for any approximation technique is to satisfy a congruity property, which implies that if the actual structure is in the class of the approximators, then the approximating network needs to match the actual one, at least in the limit of infinite data. We show that we can utilize the same distance among the nodes that is developed for recovering the structure of networks of linear dynamic systems and is estimated from blind measurements. It is shown that the computed Minimum Spanning Tree (MST) using such a distance as weights consistently recovers the undirected topology of the network when it has a polytree structure. Remarkably, this approximation algorithm is the same as the one defined in [40], which, though, was proven congruous only for rooted trees. We also provide an algorithm to congruously orient some of the links in the approximated network by extracting available features from the data. We study one interesting application of this approximation method to analyze high frequency financial market data [1].

Chapter 2

Preliminaries, Background, Assumptions and Problem Formulation

In this chapter, we provide necessary theoretical background for the problem formulation. The reader can refer to [36, 76] for most of the standard definitions in graph theory. We also mention the assumptions that are made in order to formulate the problems for which we propose a solution. More specifically, we provide definitions related to graphs with only visible nodes in Section 2.1, and definitions related to graphs containing hidden nodes in Section 2.2. Then, we introduce a class of models for linear dynamic systems in Section 2.3. Finally, in Section 2.4, we provide the formal statement of the problems that we tackle in this dissertation.

2.1 Graphs with All Visible Nodes

We recall the standard definition of directed and undirected graphs and also introduce the definition of a partially directed graph [1, 3].

Definition 2.1 (Directed and undirected graphs)**.** *A directed graph $\vec{G}$ is a pair $(N, \vec{E})$ where N is a set of nodes (or vertices) and $\vec{E}$ is a set of edges (or arcs) which are ordered pairs of elements of the set N. An undirected graph G is a pair (N, E) where N is a set of nodes (or vertices) and E is a set of edges (or arcs) which are unordered pairs of elements of the set N.* □

Definition 2.2 (Partially directed graph)**.** *A partially directed graph $\bar{G}$ is a triplet $(N, E, \vec{E})$ where N is a set of nodes, E and $\vec{E}$ are sets of undirected and directed edges, respectively, where $(y_i, y_j) \in \vec{E}$ implies $\{y_i, y_j\} \notin E$.* □

Observe that in a partially directed graph, E and $\vec{E}$ do not share any edges. We denote the unordered pair of two elements $y_i, y_j \in N$ as $y_i - y_j$ or $\{y_i, y_j\}$, and the ordered pair of $y_i, y_j \in N$ (when y_i precedes y_j) as $y_i \to y_j$ or (y_i, y_j). An example of a directed graph, an undirected graph and a partially directed graph are shown in Figures 2.1 (a) - 2.1 (c), respectively.

Furthermore, a restriction of a graph can be defined with respect to a subset of its nodes [3].

Definition 2.3 (Restriction of a graph)**.** *A directed graph $\vec{A} = (N_A, \vec{E}_A)$ is the restriction of a directed graph $\vec{G} = (N, \vec{E})$ to the nodes N_A if $N_A \subseteq N$ and $\vec{E}_A = \{(y_i, y_j) \in \vec{E} \mid y_i, y_j \in N_A\}$.* □

More informally, restriction of a graph with respect to a set of nodes A is the graph obtained by considering only the nodes in A and the edges linking pairs of nodes which are both in A.

The skeleton of a directed or partially directed graph is defined as follows [1, 2].

Definition 2.4 (Skeleton of a graph)**.** *Given a directed graph or a partially directed graph, its skeleton is the undirected graph obtained by removing the orientation from all the directed edges.*□

An example of a directed graph and its skeleton are depicted in Figure 2.1 (a) and Figure 2.1 (d), respectively. We recall the definition of degree, outdegree and indegree of a node [2].

Definition 2.5 (Degree, outdegree and indegree of a node)**.** *In a directed graph $\vec{G} = (N, \vec{E})$ or undirected graph $G = (N, E)$, degree of a vertex $y \in N$ is defined as the number of edges directly*

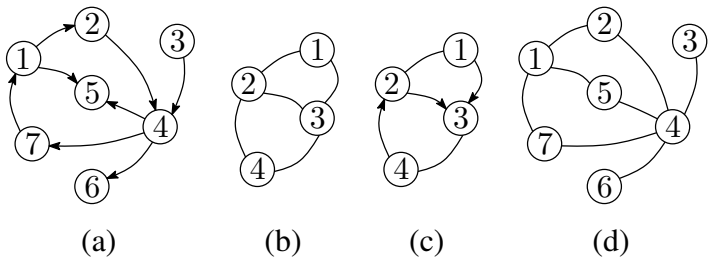


Figure 2.1: A directed graph (a), an undirected graph (b), a partially directed graph (c), and skeleton of the graph in a (d) [2, 3].

connected (or linked) to y and is denoted by $\deg_{\vec{G}}(y)$ *or* $\deg_G(y)$*, respectively. In a directed graph* $\vec{G} = (N, \vec{E})$*, outdegree of a vertex* $y \in N$ *is defined as the number of edges connected to y such that* $(y, y_i) \in \vec{E}$ *with* $y_i \in N$ *and is denoted by* $\deg^+_{\vec{G}}(y)$*. In a directed graph* $\vec{G} = (N, \vec{E})$*, indegree of a vertex* $y \in N$ *is defined as the number of edges connected to y such that* $(y_i, y) \in \vec{E}$ *with* $y_i \in N$ *and is denoted by* $\deg^-_{\vec{G}}(y)$*.* □

A root node is defined using the definition of indegree [2].

Definition 2.6 (Root node)**.** *In a directed graph* $\vec{G} = (N, \vec{E})$*, a node* $y \in N$ *is a root if* $\deg^-_{\vec{G}}(y) = 0$.□

For example, node y_3 in Figure 2.1 (a) is a root node which has $\deg(y_3) = \deg^+(y_3) = 1$ and $\deg^-(y_3) = 0$. Also, for node y_2 in the same figure, we have that $\deg(y_2) = 2$ and $\deg^+(y_2) = \deg^-(y_2) = 1$.

The definition of chain and path in a directed graph is widely used in the rest of this dissertation. In the literature of graph theory this concept is defined in a variety of different ways that are not always equivalent. Thus, we explicitly provide the definition that we use here [2].

Definition 2.7 (Path, chain and directed cycle)**.** *Consider a directed graph* $\vec{G} = (N, \vec{E})$ *where* $N = \{y_1, ..., y_n\}$*. A chain or path starting from* y_i *and ending in* y_j *is an ordered sequence of distinct edges*

$$\left((y_{\pi_0}, y_{\pi_1}), (y_{\pi_1}, y_{\pi_2}), \dots, (y_{\pi_{\ell-1}}, y_{\pi_\ell}) \right)$$

with $\ell \geq 1$ *where* $y_i = y_{\pi_0}$*,* $y_j = y_{\pi_\ell}$*, and for all* $k = 0, 1, ..., \ell - 1$ *we have* $(y_{\pi_k}, y_{\pi_{k+1}}) \in \vec{E}$ *for a chain, and either* $(y_{\pi_k}, y_{\pi_{k+1}}) \in \vec{E}$ *or* $(y_{\pi_{k+1}}, y_{\pi_k}) \in \vec{E}$ *for a path. A path in an undirected graph* $G = (N, E)$ *is the same ordered sequence where* $\{y_{\pi_k}, y_{\pi_{k+1}}\} \in E$*. When there exists at most one edge connecting each pair of nodes in* $\vec{G}$*, a path can be unambiguously determined by the sequence of nodes* $y_{\pi_0}, y_{\pi_1}, y_{\pi_2}, \dots y_{\pi_{\ell-1}}, y_{\pi_\ell}$*. Also,* ℓ *is called the length of the chain or the path. Furthermore, a directed cycle is a chain where* $y_i = y_j$*.* □

As it follows from the definition, chain is a special case of path. All paths (and consequently all chains) can be suggestively denoted by separating the nodes in the sequence $\{y_{\pi_k}\}_{k=0}^{\ell}$ with the arrow symbol $\rightarrow$ if $(y_{\pi_k}, y_{\pi_{k+1}}) \in \vec{E}$ or the symbol $\leftarrow$ if $(y_{\pi_{k+1}}, y_{\pi_k}) \in \vec{E}$. For example, in Figure 2.1 (a), the path $y_1 \rightarrow y_2 \rightarrow y_4 \rightarrow y_6$ is also a chain, while $y_1 \rightarrow y_5 \leftarrow y_4 \leftarrow y_3$ is a path, but not a chain. Furthermore, we use the symbol $-$ when we do not want to specify the orientation of the

link connecting two nodes. Consequently, it follows that the edge $\{y_i, y_j\}$ can be denoted as $y_i - y_j$ (as mentioned before) [1, 2, 3, 4]. Also, from the notion of chain, we can derive the notions of ancestors and descendants [2].

Definition 2.8 (Parent, child, ancestor and descendant)**.** *Consider a directed graph $\vec{G} = (N, \vec{E})$. A vertex y_i is a parent of a vertex y_j in $\vec{G}$ if there is a directed edge from y_i to y_j. In such a case y_j is a child of y_i in $\vec{G}$. Also y_i is an ancestor of y_j in $\vec{G}$ if either $y_i = y_j$ or there is a chain from y_i to y_j. In such a case y_j is a descendant of y_i in $\vec{G}$. Given a set $X \subseteq N$, we define the following notation:*

$$\begin{aligned}
\mathrm{pa}_{\vec{G}}(X) &:= \left\{ y_i \in N \mid \exists\, y_j \in X : y_i \textit{ is a parent of } y_j \textit{ in } \vec{G} \right\} \\
\mathrm{ch}_{\vec{G}}(X) &:= \left\{ y_j \in N \mid \exists\, y_i \in X : y_j \textit{ is a child of } y_i \textit{ in } \vec{G} \right\} \\
\mathrm{an}_{\vec{G}}(X) &:= X \cup \left\{ y_i \in N \mid \exists\, y_j \in X : y_i \textit{ is an ancestor of } y_j \textit{ in } \vec{G} \right\} \\
\mathrm{de}_{\vec{G}}(X) &:= X \cup \left\{ y_j \in N \mid \exists\, y_i \in X : y_j \textit{ is a descendant of } y_i \textit{ in } \vec{G} \right\}. \square
\end{aligned}$$

For example, node y_6 in Figure 2.1 (a) is a child of node y_4 and also a descendant of node y_3 while node y_3 is a parent of y_4 and also an ancestor of node y_6.

Given a specific path, with the exception of the first and the last nodes, we distinguish its nodes into *forks*, *inverted forks (or colliders)* and *chain links* [2].

Definition 2.9 (Fork, inverted fork and chain link)**.** *Given a path $y_{\pi_0}, ..., y_{\pi_\ell}$ in a directed graph $\vec{G} = (N, \vec{E})$, we say that y_{π_k}, for $k = 1, ..., \ell - 1$, is*

- *a fork if $(y_{\pi_k}, y_{\pi_{k-1}}) \in \vec{E}$ and $(y_{\pi_k}, y_{\pi_{k+1}}) \in \vec{E}$*
- *an inverted fork (or collider) if $(y_{\pi_{k-1}}, y_{\pi_k}) \in \vec{E}$ and $(y_{\pi_{k+1}}, y_{\pi_k}) \in \vec{E}$*
- *a chain link in all other cases.* □

For example, in Figure 2.1 (a), the path $y_7 \to y_1 \to y_5 \leftarrow y_4 \to y_6$ has a chain link in node y_1, an inverted fork in node y_5 and a fork in node y_4 [2]. Now, we can define related nodes. Informally, two nodes are related if one is a descendant of the other or if they have a common ancestor [3].

Definition 2.10 (Related nodes)**.** *Given a directed graph $\vec{G} = (N, \vec{E})$, two nodes y_i and y_j are related if there is a path connecting them that contains no inverted forks.* □

Moreover, here we provide a formal definition of polyforests, polytrees and rooted trees for completeness [1, 2, 3].

Definition 2.11 (Polyforest, polytree and rooted tree)**.** *Given a directed graph $\vec{G} = (N, \vec{E})$, $\vec{G}$ is a*

- *polyforest $\vec{F}$, if for every two nodes $y_i, y_j \in N$ there is at most one path connecting them.*
- *polytree $\vec{P}$, if for every two nodes $y_i, y_j \in N$ there is exactly one path connecting them.*
- *rooted tree $\vec{T}$, if it is a polytree with a single root.* □

Note that a rooted tree is a polytree with exactly one root. We define polytrees contained in a polyforest and also the rooted tree associated with each root of a polyforest [2, 3].

Definition 2.12 (Polytree and rooted tree of a polyforest)**.** *Each connected subgraph of a polyforest is referred to as a polytree of the polyforest. The restriction of a polyforest (polytree) to all the descendants of a root is referred to as a rooted tree of the polyforest (polytree).* □

For example, the graph in Figure 2.2 (a) is a rooted tree since it has only one root, the graph in Figure 2.2 (b) is a polytree since it has more than one root and the graph in Figure 2.2 (c) is a polyforest since it contains multiple polytrees. Note that in all of these graphs, there exists at most one path connecting any pairs of nodes. Also, observe that the polytree of Figure 2.2 (b) contains three rooted trees and the polyforest of Figure 2.2 (c) contains two polytrees.

The following proposition guarantees that in a rooted tree there are no paths with inverted forks [2]. In other words, all the nodes in a rooted tree are related.

Proposition 2.13. *In a rooted tree, there are no paths containing inverted forks.*

Proof. The proof is left to the reader. □

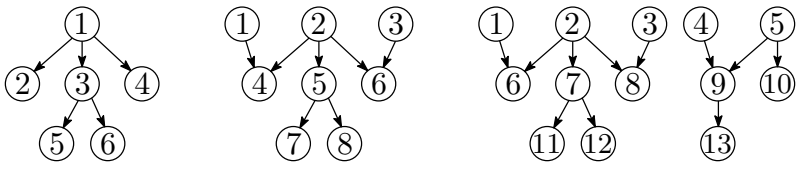


Figure 2.2: A rooted tree graph (a), a polytree graph (b), and a polyforest graph (c).

2.2 Graphs with Hidden Nodes

In this section we provide the necessary background for dealing with graphs with latent (or hidden) nodes. Latent graphs are an extension of standard graphs which were discussed in previous section (also see [53] for an equivalent definition) [2, 3].

Definition 2.14 (Latent graph)**.** *We define a directed latent graph $\vec{G}_\ell$ as a triplet $(V, L, \vec{E})$ such that V (the set of visible nodes) and L (the set of hidden or latent nodes) are disjoint, and $\vec{G} = (N, \vec{E})$ is a directed graph where $N := V \cup L$. Also, we say that $\vec{G}_\ell$ is a latent rooted tree, a latent polytree or a latent polyforest if $\vec{G}$ is respectively a rooted tree, a polytree, or a polyforest.* □

Observe that the notation used for latent graphs should not be confused with the notation used for partially directed graphs since they have different element sets as their triplets. As an example of latent graphs, the graph shown in Figure 2.3 (a) is a latent graph where its latent nodes, node y_2 and node y_5, are shown by dotted circles.

We can extend the definition of a partially directed graph to latent partially directed graph considering a partition of the set of nodes into visible and hidden nodes [4].

Definition 2.15 (Latent partially directed graph)**.** *A latent (or hidden) partially directed graph $\bar{G}_\ell$ is a 4-ple $(V, L, E, \vec{E})$ where*

- *the disjoint sets V and L are named the set of visible nodes and the set of hidden nodes,*
- *the set E is the set of undirected edges containing unordered pairs of $(V \cup L) \times (V \cup L)$,*
- *the set $\vec{E}$ is the set of directed edges containing ordered pairs of $(V \cup L) \times (V \cup L)$.*

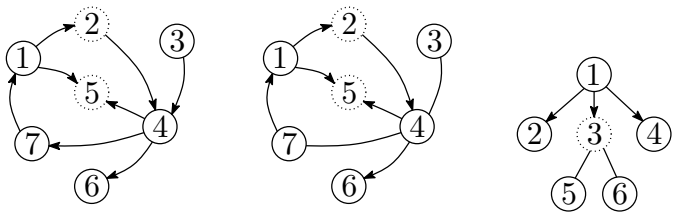


Figure 2.3: A latent graph (a), a latent partially directed graph (b), and a latent partially directed tree (c).

In a latent partially directed graph the sets E and $\vec{E}$ do not share any edges. Namely, $y_i - y_j \in E$ implies that both $y_i \to y_j$ and $y_j \to y_i$ are not in $\vec{E}$. □

Figure 2.3 (b) is an example of a latent partially directed graph. A latent partially directed graph is a fully undirected latent graph when $\vec{E} = \emptyset$, and we simplify the notation by writing $G_\ell = (V, L, E)$. Similarly, when $E = \emptyset$, we have a fully directed latent graph, and we denote it by $\vec{G}_\ell = (V, L, \vec{E})$. Observe that, if we drop the distinction between visible and hidden nodes and consider $V \cup L$ as the set of nodes, we recover the standard notions of undirected and directed graphs. Thus, latent partially directed graphs inherit, in a natural way, all notions associated with standard graphs as discussed in the previous section (e.g., paths, degree, parents, children, etc.). Consequently, we can define restriction of a latent partially directed graph similar to Definition 2.3. In a similar way, when every two nodes in a latent partially directed graph can be connected through exactly one path we have a latent partially directed tree [4].

Definition 2.16 (Latent partially directed tree)**.** *A latent partially directed graph $\bar{G}_\ell = (V, L, E, \vec{E})$ is a latent partially directed tree when every pair of nodes $y_i, y_j \in V \cup L$ is connected by exactly one path.* □

Figure 2.3 (c) is an example of a latent partially directed tree. Trivially, latent partially directed trees generalize the notions of undirected trees and polytrees (directed trees) [68]. In a latent partially directed tree, we define a hidden cluster as a group of hidden nodes connected to each other via a path constituted exclusively of hidden nodes [4].

Definition 2.17 (Hidden cluster)**.** *A hidden cluster in a latent partially directed tree $\vec{P}_\ell = (V, L, E, \vec{E})$ is a set $C \subseteq L$ such that for each distinct pair of nodes $y_i, y_j \in C$ the unique path connecting them contains only nodes in C and no node in C is linked to a node which is in $L \setminus C$.* □

Figure 2.4 (a) depicts a latent partially directed tree (actually a latent polytree). In this figure, the hidden clusters C_1 and C_2 are highlighted by the red dotted lines. Observe that each node in a hidden cluster has neighbors which are either visible or hidden nodes of the same cluster. Therefore, we introduce the set of (visible) neighbors of a hidden cluster [4].

Definition 2.18 (Neighbors, closure and degree of a hidden cluster)**.** *In a latent partially directed tree, the set of all visible nodes linked to any of the nodes of a hidden cluster C is the set of*

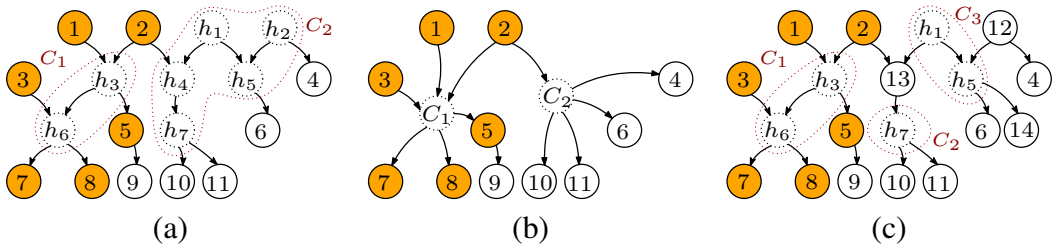


Figure 2.4: A generic hidden polytree (a) and its collapsed hidden polytree (b), a minimal hidden polytree (c) [4].

neighbors of C and is denoted by $N(C)$. We define the degree of the cluster as $|N(C)|$, namely the number of neighbors of the cluster. We refer to the restriction of a latent polytree to a hidden cluster and its neighbors as the closure of the hidden cluster. □

Consider again the latent polytree of Figure 2.4 (a). The neighbors of the hidden cluster C_1 are y_1, y_2, y_3, y_5, y_7 and y_8 (also highlighted with orange color). We also define the notion of root of a hidden cluster [4].

Definition 2.19 (Root of a hidden cluster in a latent polytree)**.** *Let $\vec{P}_\ell = (V, L, \vec{E})$ be a latent polytree. Any root of the restriction of $\vec{P}_\ell$ to one of the hidden clusters of $\vec{P}_\ell$ is called a root of the hidden cluster.* □

In the latent polytree of Figure 2.4 (a), the node y_{h_3} is the hidden root of C_1 and the nodes y_{h_1} and y_{h_2} are the hidden roots of C_2. Observe that a hidden cluster might have multiple hidden roots.

Given a latent partially directed tree, we can define its *collapsed representation* by replacing each hidden cluster with a single hidden node. The formal definition is as follows [4].

Definition 2.20 (Collapsed representation)**.** *We define the collapsed representation of $\vec{P}_\ell = (V, L, E, \vec{E})$ as the latent partially directed tree $\vec{P}_c = (V, L_c, E_c, \vec{E}_c)$. Let n_c be the number of hidden clusters $C_1, ..., C_{n_c}$ and let $L_c = \{C_1, ..., C_{n_c}\}$, and*

$$E_c := \{y_i - y_j \in E \mid y_i, y_j \in V\} \cup \{y_i - C_k \mid \exists y_j \in C_k, y_i - y_j \in E\} \cup \{C_k - y_j \mid \exists y_i \in C_k, y_i - y_j \in E\}$$

$$\vec{E}_c := \{y_i \to y_j \in \vec{E} \mid y_i, y_j \in V\} \cup \{y_i \to C_k \mid \exists y_j \in C_k, y_i \to y_j \in \vec{E}\} \cup \{C_k \to y_j \mid \exists y_i \in C_k, y_i \to y_j \in \vec{E}\}.$$ □

As an example, the collapsed representation of the latent polytree in Figure 2.4 (a) is depicted in Figure 2.4 (b).

In the next chapters, we will show in what cases graphical models with polytree structures can be recovered from the independence relations involving only visible nodes. Specifically, we assume that a polytree is a perfect map (see [36, 77]) for a probabilistic model defined over the variables $V \cup L$ where V and L are disjoint sets. We will find conditions under which it is possible to recover information about the perfect map of the probabilistic model considering only independence relations of the form $\mathcal{I}(y_i, \emptyset, y_j)$ (read y_i and y_j are conditionally independent) and of the form $\mathcal{I}(y_i, y_k, y_j)$ (read y_i and y_j are conditionally independent given y_k) for all visible nodes $y_i, y_j, y_k \in V$ [4].

One of the fundamental requirements is that all hidden nodes need to satisfy certain degree conditions summarized in the following definition [4].

Definition 2.21 (Minimal latent polytree)**.** *A latent polytree $\vec{P}_\ell = (V, L, \vec{E})$ is minimal if every hidden node node $y_h \in L$ satisfies one of the following conditions:*

- $\deg^+_{\vec{P}_\ell}(y_h) \geq 2$ *and* $\deg_{\vec{P}_\ell}(y_h) \geq 3$ *and if* $|\mathrm{pa}_{\vec{P}_\ell}(y_h)| = 1$, *then* $\mathrm{pa}_{\vec{P}_\ell}(y_h) \subseteq V$;
- $\deg^+_{\vec{P}_\ell}(y_h) = 2$ *and* $\deg^-_{\vec{P}_\ell}(y_h) = 0$ *and* $\deg^-_{\vec{P}_\ell}(y_{c_1}), \deg^-_{\vec{P}_\ell}(y_{c_2}) \geq 2$ *where* $\mathrm{ch}_{\vec{P}_\ell}(y_h) = \{y_{c_1}, y_{c_2}\}$. □

Note that the nodes y_{h_2}, y_{h_4}, y_{h_5}, y_{h_7} in Figure 2.4 (a) do not satisfy the minimality conditions and therefore the hidden polytree is not minimal. Instead, Figure 2.4 (c) shows a minimal latent polytree. We also define two special types of hidden nodes in a latent polytree and in the next chapters we explain why we need to make this distinction [4].

Definition 2.22 (Type-I and Type-II hidden nodes)**.** *In a minimal polytree, we call a hidden node y_h where $\deg_{\vec{G}}(y_h) = 2$ with at least one visible child, a Type-II hidden node. All other hidden nodes are Type-I hidden nodes.* □

In the minimal latent polytree of Figure 2.5 (a), the nodes y_{h_2} and y_{h_3} are Type-II hidden nodes, while all the other hidden nodes are Type-I. In order to deal with Type-II hidden nodes separately, as explained in the next chapters, we define the quasi-skeleton of a minimal polytree [4].

Definition 2.23 (Quasi-skeleton of a minimal latent polytree)**.** *Let $\vec{P}_\ell = (V, L, \vec{E})$ be a minimal latent polytree. The quasi-skeleton of $\vec{P}_\ell$ is the undirected graph obtained by removing all the*

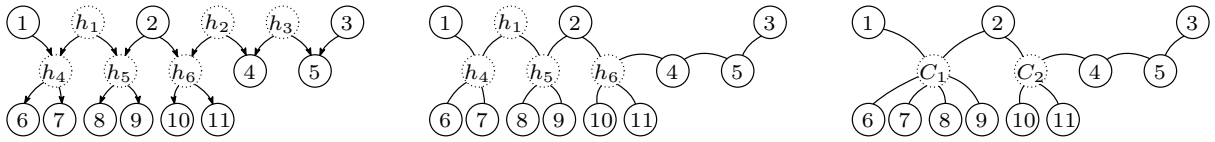


Figure 2.5: A minimal latent polytree $\vec{P}_\ell$ containing Type-I and Type-II hidden nodes (a), quasi-skeleton of $\vec{P}_\ell$ (b), and collapsed quasi-skeleton of $\vec{P}_\ell$ (c) [4].

orientation of edges in $\vec{P}_\ell$ and also removing all the Type-II hidden nodes and then linking their two children together. □

For example, the quasi-skeleton of the polytree of Figure 2.5 (a) is depicted in Figure 2.5 (b). Observe that in the quasi-skeleton of $\vec{P}_\ell$, Type-II hidden nodes have been eliminated and their children are linked together. Furthermore, observe that we can obtain the collapsed quasi-skeleton of a polytree by replacing the hidden clusters with individual hidden nodes in the quasi-skeleton of a polytree as depicted in Figure 2.5 (c).

As it is well known in the theory of graphical models, in the general case, from a set of conditional independence statements (formally, a semi-graphoid) faithful to a DAG, it is not possible to recover the full DAG [37, 62]. What can be recovered for sure is the pattern of the DAG, namely the skeleton and the v-structures (i.e., $y_i \rightarrow y_k \leftarrow y_j$ or the inverted forks) of the DAG [37, 62]. In the next chapters, we will show that, similarly, in the case of a minimal latent polytree, we are able to recover the pattern of the polytree from the independence statements involving only the visible variables [4]. The following is a formal definition of the pattern of a polytree (also see [62]).

Definition 2.24 (Pattern of a polytree)**.** *Let $\vec{P} = (N, \vec{E})$ be a polytree. The pattern of $\vec{P}$ is a partially directed graph where the orientation of all the v-structures (i.e., $y_i \rightarrow y_k \leftarrow y_j$) are known and as many as the remaining undirected edges are oriented in such a way that the other alternative orientation would result in a v-structure.* □

2.3 Linear Dynamic Influence Models

In this section we introduce a Linear Dynamic Influence Model (LDIM) and some related properties. An LDIM (see the equivalent definition of Linear Dynamic Graph in [2, 19]) is a class of models describing a network of dynamic systems. We assume that the dynamics of the nodes in the network are represented by scalar random processes $\{y_i\}_{i=1}^n$. Each process is given by the superposition of an independent component (or input) u_i and the influences coming from its parent nodes through dynamic links. The unknown input acting on each node is modeled as noise and is assumed to be uncorrelated with other inputs. Namely, we have that the power spectral density $\Phi_{u_i u_j}(z) = 0$ which we also denote as $u_i \perp\!\!\!\perp u_j$. If a certain process directly influences another one, then a directed edge is drawn between them and as a result a directed graph is obtained. In a more informal way, an LDIM is a network of stochastic processes $y_1, ..., y_n$ interconnected with each other via input/output relations defined by the transfer functions populating the transfer matrix $H(z)$. The formal definition of an LDIM and its associated graph are as follows [1, 3].

Definition 2.25 (Linear Dynamic Influence Model and its associated graph)**.** *A Linear Dynamic Influence Model (LDIM) is defined as a pair $\mathcal{G} = (H(z), u)$ where*

- *$u = (u_1, ..., u_n)^T$ is a vector of n wide-sense stationary stochastic processes with finite variance such that $\Phi_u(z)$, the power spectral density of u, is rational and diagonal; and*
- *$H(z)$ is an $n \times n$ matrix of rational, causal and stable transfer functions with $H_{ij}(z)$ being the entry (i, j) of $H(z)$ such that $H_{ii}(z) = 0$ for $i = 1, ..., n$.*

The output processes $\{y_i\}_{i=1}^n$ of the LDIM are defined as

$$y_i = u_i + \sum_{j=1}^{n} H_{ij}(z) y_j \tag{2.1}$$

or in a more compact way as $y = u + H(z)y$. We define the associated graph of the LDIM as $\vec{G} = (N, \vec{E})$ where $N = \{y_1, ..., y_n\}$ and $\vec{E} = \{(y_i, y_j) \mid H_{ji}(z) \neq 0\}$. When the associated graph of an LDIM is a rooted tree, or a polytree, or a polyforest, we call it a Linear Dynamic Rooted Tree (LDRT), or a Linear Dynamic Polytree (LDPT), or a Linear Dynamic Polyforest (LDPF), respectively. □

As mentioned before, we are interested in studying networks when not all of the nodes are measurable. Thus, when only a subset of the nodes of an LDIM is observable, we define the associated latent graph of an LDIM with respect to the observed nodes [3].

Definition 2.26 (Associated latent graph of an LDIM)**.** *Given that only a subset V of N of the associated graph of an LDIM is known, the LDIM is a latent LDIM and its associated graph is a latent graph denoted by $\vec{G}_\ell = (V, L, \vec{E})$ where V is the set of visible nodes and L is the set of hidden nodes and V and L are disjoint.* □

Lemma 2.27 guarantees the well-posedness of LDIMs with tree structure [3].

Lemma 2.27. *Let $\mathcal{G} = (H(z), u)$ be an LDIM with associated graph $\vec{G} = (N, \vec{E})$ and $y_i, y_j \in N$. Assume that there are no directed cycles and that $\ell < +\infty$ is the length of the longest chain in $\vec{G}$. Then, we have that*

$$y = T(z)u := \left(\mathbf{I} + \sum_{k=1}^{\ell} H^k(z)\right)u \tag{2.2}$$

with $T_{ii}(z) = 1$ for all i. Also, if there is no chain from y_i to y_j where $y_j \neq y_i$, we have $T_{ji}(z) = 0$ and $\Phi_{y_j u_i}(z) = 0$.

Proof. The proof is in Appendix A.1. □

Now that we have defined LDIMs, we define the following distance on their nodes. In the next chapters we show how we can leverage this distance for the learning of polyforest networks [2, 3].

Definition 2.28 (Log-coherence distance)**.** *Given an LDIM with nodes $\{y_1, ..., y_n\}$, we define the log-coherence distance*

$$d_L\left(y_i, y_j\right) = \int_{-\pi}^{\pi} -\log|C_{ij}(e^{\mathbf{i}\omega})|d\omega \tag{2.3}$$

with $C_{ij}(e^{\mathbf{i}\omega}) = \frac{|\Phi_{y_i y_j}(e^{\mathbf{i}\omega})|^2}{\Phi_{y_i}(e^{\mathbf{i}\omega})\Phi_{y_j}(e^{\mathbf{i}\omega})}$, which is commonly known as the coherence between the signals y_i and y_j, where $\Phi_{y_i}(e^{\mathbf{i}\omega})$ and $\Phi_{y_j}(e^{\mathbf{i}\omega})$ are the spectral densities of y_i and y_j, respectively and $\Phi_{y_i y_j}(e^{\mathbf{i}\omega})$ is the cross-spectral density of y_i and y_j. □

Furthermore, we will show that the property of topological identifiability enables the reconstruction of an LDPF from data. This property means that the distance between any two linked nodes in the associated graph of an LDIM has a finite and non-zero value. Here, we formally introduce a topologically identifiable LDIM [3].

Definition 2.29 (Topological identifiability)**.** *Let $\vec{G} = (N, \vec{E})$ be the associated graph of the LDIM $\mathcal{G}$ with a tree skeleton. $\mathcal{G}$ is topologically identifiable if for every edge $(y_i, y_j) \in \vec{E}$, we have that $0 < d_L\left(y_i, y_j\right) < \infty$.* □

The following lemma shows that mild conditions are required to guarantee the topological identifiability property [3].

Lemma 2.30. *Let $\mathcal{G} = (H(z), u)$ be an LDIM where $z = e^{\mathbf{i}\omega}$. $\mathcal{G}$ is topologically identifiable if $\Phi_{u_i}(e^{\mathbf{i}\omega}) > \eta > 0$ for some η, every ω and all i, and each entry of $H(z)$ that is not identically null has no zeros on the unit circle.*

Proof. The proof is in Appendix A.2. □

Note that the mild conditions of Lemma 2.30 can be further relaxed. Given that the contiguous nodes have finite non-zero distance, the property of additivity along the paths of a graph, as defined in the following definition, allows us to extend Lemma 2.30 to all the nodes in the same LDRT [3].

Definition 2.31 (Additivity of a distance along the paths)**.** *Let $\vec{G} = (N, \vec{E})$ be a directed graph. The distance $d(y_i, y_j)$ is additive along the paths in $\vec{G}$ if y_k on the path from node y_i to node y_j implies $d(y_i, y_j) = d(y_i, y_k) + d(y_k, y_j)$.* □

The notion of *d-separation* (see [35]) will play an important role to prove that the distance in Equation (2.3) is additive along the paths of an LDRT.

Definition 2.32 (d-separation)**.** *Let $\vec{G} = (N, \vec{E})$ be a directed graph. We say that the nodes $y_i, y_j \in N$ are d-separated by the set K in $\vec{G}$ where $K \subseteq N$ if at least one of the following conditions is true:*

- *if $\exists y_k \in K$ on all of the paths from y_i to y_j such that y_k is a chain or a fork.*
- *if $\exists y_k \notin K$ on all of the paths from y_i to y_j such that y_k is an inverted fork and $\mathrm{de}_{\vec{G}}(y_k) \cap K = \emptyset$.*

We use $dsep < y_i, K, y_j >_{\vec{G}}$ to denote d-separation of y_i from y_j by the set K in graph $\vec{G}$. □

Note that d-separation is a notion defined in general for DAGs, however, in graphs with polyforest structures it is easier to check the conditions since there exists at most one path between any pair of nodes. For example, it is immediate to check that in the polyforest of Figure 2.1 (a) we have that $dsep \prec y_2, \{y_1, y_4\}, y_7 \succ_{\vec{F}}$ but not $dsep \prec y_2, \{y_4\}, y_3 \succ_{\vec{F}}$.

The following result states that the log-coherence distance of Equation (2.3) is additive along the paths of an LDRT [3].

Proposition 2.33. *Let $\mathcal{T} = (H(z), u)$ be an LDRT with the associated tree graph $\vec{T} = (N, \vec{E})$. The log-coherence distance $d_L\left(y_i, y_j\right)$ for all $y_i, y_j \in N$ is additive along the paths of $\vec{T}$.*

Proof. The proof is in Appendix A.3. □

Finally, we provide the following important characterization which states that in an LDPF the log-coherence distance of two related nodes is finite and non-zero [3].

Proposition 2.34. *Let $\mathcal{F} = (H(z), u)$ be a topologically identifiable LDPF with associated graph $\vec{F} = (N, \vec{E})$. Let y_i, y_j be distinct nodes in N. There exists a path from y_i to y_j with no inverted fork (i.e., y_i and y_j are related) if and only if $0 < d_L\left(y_i, y_j\right) < \infty$.*

Proof. The proof is in Appendix A.4. □

2.4 Problem Statement

After reviewing the necessary background and stating the assumptions, here we propose the formal statement of the problems that we tackle in this dissertation.

The first contribution of this dissertation considers an LDPF with output processes $\{y_i\}_{i=1}^n$, assuming that only (cross-)spectral densities $\Phi_{y_i y_j}(e^{\mathbf{i}\omega})$ of a subset V of the processes (signals) are known. We want to determine if there exists an edge linking any two processes y_i and y_j, and also find the orientation of the recovered links by extracting available features from the data or exploiting some a priori knowledge [3]. It is noteworthy that one of the important features of the method developed for solving this problem is that it can be applied to networks of random variables with virtually no modifications.

As another contribution of this dissertation, we would like to propose a solution to the following problem. Assume a semi-graphoid defined over the visible and hidden variables $V \cup L$. Let the latent polytree $\vec{P}_\ell = (V, L, \vec{E})$ be faithful to this semi-graphoid. The goal is to recover the pattern of $\vec{P}_\ell$ only from the information obtained by observing the visible nodes. Our proposed approach makes use only of the conditional independence relations of the form $\mathcal{I}(y_i, \emptyset, y_j)$ or $\neg\mathcal{I}(y_i, \emptyset, y_j)$, and $\mathcal{I}(y_i, y_k, y_j)$ or $\neg\mathcal{I}(y_i, y_k, y_j)$ for all the visible nodes $y_i, y_j, y_k \in V$. In other words, this method only makes use of the third order statistics of the observed nodes to recover the pattern of the latent polytree [4].

Another contribution of this dissertation is proposing an algorithm for approximating a general network with a simpler structure such as a polytree network when only observations of the node processes are given. This approximation is motivated by following a form of Occam's razor principle since the approximating polytree is optimum in the sense that it has fewer number of edges compared to the original graph. It is noteworthy to mention that this approximation method exploits the assumption of linearity and assumes that all the nodes in the system are observable [1].

Chapter 3

Learning Linear Networks with Tree Structures

In this chapter, we first recall an algorithm from the literature that learns the skeleton of a rooted tree when some nodes in the network are not measurable. We then show its limitations on learning the skeleton of polyforests. In the following sections, we present a new algorithm that is capable of learning the skeleton of polyforest networks. We also present an algorithm that recovers the orientation of some of the links in the skeleton of the recovered polyforest. Finally, we provide the fundamental limitations for solving the problem of learning polyforest networks with the presence of hidden nodes.

3.1 Reconstruction of Rooted Trees with Hidden Nodes via An Additive Metric

Recursive Grouping Algorithm (RGA) is an enabling and computationally efficient result for identification of an undirected tree structure in the presence of unobservable nodes [53]. RGA achieves this task so long as every hidden node in the graph has degree greater than or equal to 3 and a distance is defined among the nodes such that it is additive along every path of the tree graph. The only information required by RGA is the distance between each pair of visible nodes [2, 3]. Also, note that RGA can only be applied to networks of random variables.

The following Algorithm 1 is an equivalent but simplified version of RGA that is reported here for completeness and illustrative purposes, only.

Algorithm 1 Simplified Recursive Grouping Algorithm

Input V and the distances $d(y_i, y_j)$ for $y_i, y_j \in V$
Output $T = (V \cup L, E)$ and the distances $d(y_i, y_j)$ for $y_i, y_j \in V \cup L$

Define $d_{ij} := d(y_i, y_j)$ for $y_i, y_j \in V$
Initialize $Y := V$, $L := \emptyset$, $E := \emptyset$ and $c := 0$
Define $\Phi_{ijk} := d_{ik} - d_{jk}$ for distinct $y_i, y_j, y_k \in Y$
for each pair of distinct nodes $y_i, y_j \in Y$ **do**
    if $\Phi_{ijk} = d_{ij}$ for all $y_k \in Y \setminus \{y_i, y_j\}$ **then**
        y_i is a leaf and y_j is its parent
        set $E := E \cup \{\{y_i, y_j\}\}$ and $Y := Y \setminus \{y_i\}$
    end if
    if $\Phi_{ijk} = -d_{ij}$ for all $y_k \in Y \setminus \{y_i, y_j\}$ **then**
        y_j is a leaf and y_i is its parent
        set $E := E \cup \{\{y_i, y_j\}\}$ and $Y := Y \setminus \{y_j\}$
    end if
    if $-d_{ij} < \Phi_{ijk} = \Phi_{ijk'} < d_{ij}$ for all $y_k, y_{k'} \in Y \setminus \{y_i, y_j\}$ **then**
        y_i and y_j are leaves and they are siblings
        compute $q_k = \frac{d_{jk}+d_{ik}-d_{ij}}{2}$
        if $q_k \neq 0$ for all $y_k \in Y \setminus \{y_i, y_j\}$ **then**
            consider y_{h_c} as a new hidden node
            set $Y := Y \setminus \{y_i, y_j\}$, $L := L \cup \{y_{h_c}\}$ and $E := E \cup \{\{y_i, y_{h_c}\}, \{y_j, y_{h_c}\}\}$
            add the distances $d_{h_c k} := q_k$ for $y_k \in Y$ and $d_{h_c i} = d_{h_c j} := \frac{1}{2}(d_{ij} + \Phi_{ijk})$
            set $c := c + 1$
        end if
        if $q_k = 0$ for some $y_k \in Y \setminus \{y_i, y_j\}$ **then**
            y_k is the parent of the leaves y_i and y_j
            set $Y := Y \setminus \{y_i, y_j\}$ and $E := E \cup \{\{y_i, y_k\}, \{y_j, y_k\}\}$
        end if
    end if
end for
Define $d(y_i, y_j) := d_{ij}$ for $y_i, y_j \in V \cup L$

In order to show the limitations of RGA when multiple roots are present, we differentiate between two types of identifiable hidden nodes [2, 3].

Definition 3.1 (RGA-detectable and FD-detectable hidden nodes)**.** *Let* $\vec{F}_\ell = (V, L, \vec{E})$ *be a latent polyforest. A hidden node* $y_h \in L$ *is*

- *RGA-detectable if* $\deg^+_{\vec{F}_\ell}(y_h) \geq 2$ *and* $\deg_{\vec{F}_\ell}(y_h) \geq 3$,

- *Finite-Distance detectable (FD-detectable) if* $\deg^+_{\vec{F}_\ell}(y_h) = 2$, $\deg^-_{\vec{F}_\ell}(y_h) = 0$, $\deg^-_{\vec{F}_\ell}(y_{c_1}) \geq 2$ *and* $\deg^-_{\vec{F}_\ell}(y_{c_2}) \geq 2$ *where* $\mathrm{ch}_{\vec{F}_\ell}(y_h) = \{y_{c_1}, y_{c_2}\}$. □

Observe that these two types of hidden nodes are not exhaustive and in Section 3.4 we show that it is not possible to detect any other type of hidden nodes under the specified assumptions [2, 3].

In the case of an LDPF $\vec{F}$, we have shown that the log-coherence distance $d_L\left(y_i, y_j\right)$ has the property of being additive only along the paths of each rooted tree of $\vec{F}$ (see Proposition 2.33). Thus, if $\vec{F}$ has a unique root, namely $\vec{F}$ is a rooted tree, RGA can be applied to reconstruct its skeleton. The following theorem formalizes this idea [2, 3].

Theorem 3.2. *Let* $\vec{T} = (V, L, \vec{E})$ *be the associated graph of a topologically identifiable latent LDRT. Let all hidden nodes* $y_h \in L$ *be RGA-detectable and let the visible nodes set* V *and the distances* $d_L\left(y_i, y_j\right)$ *for* $y_i, y_j \in V$ *be the input of RGA. Then, the output of RGA is the skeleton of* $\vec{T}$.

Proof. The proof is in Appendix B.1. □

However, Proposition 2.33 does not hold for LDPTs or LDPFs which are more general classes of networks [68]. As an example, consider the associated graph of an LDPT with only 3 nodes as depicted in Figure 3.1. From Proposition 2.34, we have $d_L(y_1, y_2) = \infty$. On the other hand, if the additive property held, we would have $d_L(y_1, y_2) = d_L(y_1, y_3) + d_L(y_3, y_2)$ which implies that $d_L(y_1, y_2) < \infty$ because $d_L(y_1, y_3)$ and $d_L(y_3, y_2)$ are positive finite values. Thus, this example illustrates that it is not possible to extend the additive property of the distance $d_L\left(y_i, y_j\right)$ to trees with multiple roots [3].

A possible idea would be, given the visible nodes of a polyforest, to find a way to determine all visible nodes that belong to the same rooted tree $\vec{T}_\ell$. The additivity property of the distance would be satisfied on the nodes in each rooted tree $\vec{T}_\ell$, allowing to apply RGA. However, even in this

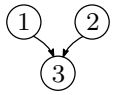


Figure 3.1: The graph associated with a sample LDPT

case RGA would fail to correctly reconstruct the skeleton of $\vec{T}_\ell$ in the presence of FD-detectable hidden nodes. More formally, the following proposition proves that when there exists at least one FD-detectable hidden node in a polyforest, RGA would fail to correctly reconstruct the skeleton of its rooted trees [3].

Proposition 3.3. *Let $\vec{T}_\ell = (V_T, L_T, \vec{E}_T)$ be the associated graph of a rooted tree of a topologically identifiable LDPF with the associated graph $\vec{F}_\ell$ where $\exists y_h \in L_T$ that is FD-detectable and all the other hidden nodes are RGA-detectable. The output of RGA applied to the distances $d_L\left(y_i, y_j\right)$ for all $y_i, y_j \in V_T$ is the tree $T_X = (V_X \cup L_X, E_X)$ with $V_X = V_T$, $L_X = L_T \setminus \{y_h\}$, and $E_X = \{\{\{y_i, y_j\} \mid (y_i, y_j) \in \vec{E}_T\} \cup \{y_{c_1}, y_{c_2}\}\} \setminus \{\{y_h, y_{c_1}\}, \{y_h, y_{c_2}\}\}$ where $\mathrm{ch}_{\vec{F}_\ell}(y_h) = \{y_{c_1}, y_{c_2}\}$.*

Proof. The proof is in Appendix B.2. □

We will show this result by an example. Consider the associated graph of an LDPT system containing both FD-detectable and RGA-detectable hidden nodes as shown in Figure 3.2 (a). Figure 3.2 (b) shows the visible descendants of the FD-detectable hidden node y_{h_2} and Figure 3.2 (c) illustrates the output of RGA when applied to the visible descendants of the FD-detectable hidden node, y_{h_2}. Note that y_{h_1} is RGA-detectable and RGA has correctly identified this hidden node, however, the FD-detectable hidden node y_{h_2} is not detected by RGA [3].

In the next section, we propose an algorithm capable of identifying both RGA-detectable and FD-detectable hidden nodes.

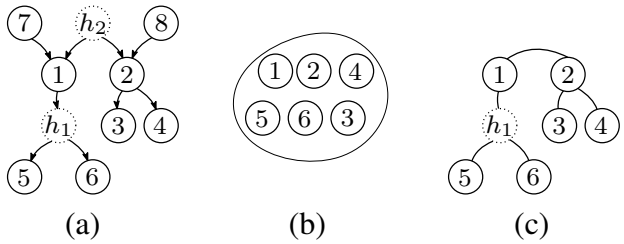


Figure 3.2: A polyforest with one FD-detectable hidden node, y_{h_2}, and one RGA-detectable hidden node, y_{h_1} (a), set of visible descendants of y_{h_2} (b), and output of the application of RGA (c). RGA detects the RGA-detectable hidden node, y_{h_1}, but incorrectly connects the children of the FD-detectable hidden node, y_{h_2}, as shown in Proposition 3.3 [3].

3.2 An Algorithm to Learn Latent Polyforest Networks

The methods presented here, as opposed to RGA, will be shown capable of identifying the structure of a polyforest network of dynamic systems when each hidden node is either RGA-detectable or FD-detectable. This motivates the following definition [3].

Definition 3.4 (Structural identifiability)**.** *A latent polyforest* $\vec{F}_\ell = (V, L, \vec{E})$ *is structurally identifiable if every hidden node* $y_h \in L$ *is either RGA-detectable or FD-detectable. By extension, a latent LDPF is structurally identifiable if its associated graph is structurally identifiable.* □

Given the power spectral and cross-spectral densities of the visible nodes V of a latent LDPF with the associated graph of $\vec{F}_\ell = (V, L, \vec{E})$, in order to learn the structure of $\vec{F}_\ell$, we follow four main steps [3]:

A. Obtain the lists of visible nodes corresponding to each rooted tree in $\vec{F}_\ell$;

B. For each list obtained at Step A, identify the skeleton of the rooted tree;

C. Merge the subgraphs obtained at Step B, considering the potential presence of overlap between rooted trees in the original polyforest;

D. Identify the link orientations of the skeleton.

Figure 3.3 (True) illustrates an example of a polytree graph associated with an LDPT and the aforementioned four steps are shown in Figures 3.3 (Step A)-(Step D). We discuss these steps in details in the following subsections [3].

3.2.1 Step A. Obtain the Visible Descendants of Each Root

In this section we introduce Pairwise-Finite Distance Algorithm (PFDA), presented in Algorithm 2, which outputs the sets of visible descendants of each root in a polyforest. Hypothetically, if the structure of the polyforest to reconstruct were known a priori (including its hidden nodes), it would be trivial to identify all the roots and their visible descendants. However, since the goal is to precisely infer the structure of the polyforest only from the knowledge of observable processes, we need to have an algorithm that requires neither the knowledge of the structure nor any information

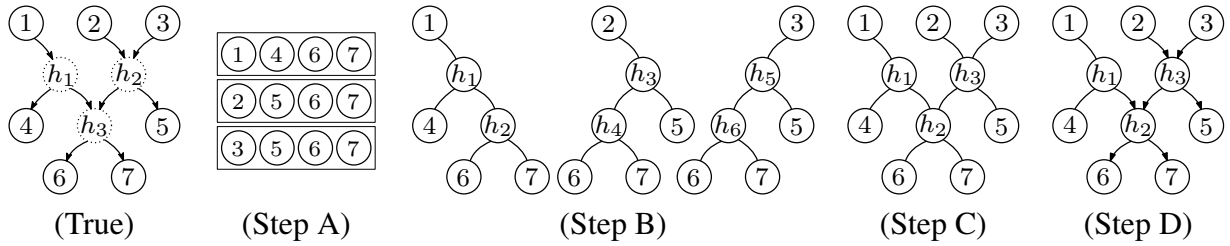


Figure 3.3: The polytree graph associated with an LDPT (True), output of Step A is the set of lists of the visible nodes in each rooted tree (Step A), output of Step B is the skeleton of each rooted tree (Step B), output of Step C is the skeleton of the polytree (Step C), and output of Step D is the partially oriented polytree (Step D) [3].

about the hidden nodes (some of which could even be roots). PFDA takes an ordered list of visible nodes V and their distances as input. Then for each pair of distinct nodes $y_i, y_j \in V$ such that $d(y_i, y_j) < \infty$, it initializes an unordered list $S_{i,j}$ with $\{y_i, y_j\}$ and proceeds by adding elements to the list so long as the added element has a finite distance to all the elements already in $S_{i,j}$. The output of PFDA is given by all the distinct lists $S_{i,j}$, for $i \neq j$, where each list represents the observable nodes in a rooted tree of the polyforest [3].

Algorithm 2 Pairwise-Finite Distance Algorithm

Input the ordered set of nodes $V = \{y_1, ..., y_n\}$ and the distances $d(y_i, y_j)$ for $y_i, y_j \in V$
Output the set of all non-eliminated lists $S_{i,j}$

for every node $y_i \in V$ such that $\forall y_j \in V \setminus \{y_i\}$ we have that $d(y_i, y_j) = \infty$ **do**
    define $S_{i,0} := \{y_i\}$
end for
for each pair $y_i, y_j \in V$ with $i < j$, and $d(y_i, y_j) < \infty$ **do**
    define $S_{i,j} := \{y_i, y_j\}$
    for each $y_k \in V \setminus S_{i,j}$ **do**
        if $\forall y \in S_{i,j} : d(y_k, y) < \infty$, then add y_k to $S_{i,j}$
    end for
end for
for each pair $y_i, y_j \in V$ with $i < j$ **do**
    if $S_{i,j} = S_{k,\ell}$ for some k and ℓ, then eliminate $S_{k,\ell}$
end for

It is straightforward to conclude that the time complexity of PFDA is upper-bounded by a quartic polynomial in the number of visible nodes in the worst case scenario. Observe that PFDA requires an ordering of V and thus its output, in general, might depend on such an ordering. A first

enabling result is that, irrespective of the ordering on V, every list returned by PFDA corresponds to the visible descendants of a root in the polyforest [3].

Theorem 3.5. *Let $\vec{F}_\ell = (V, L, \vec{E})$ be the associated graph of a latent LDPF and define an arbitrary ordering on $V = \{y_1, ..., y_n\}$. Let $d\left(y_i, y_j\right)$ be a distance defined on V such that $d\left(y_i, y_j\right) < \infty$ if and only if y_i and y_j are related. Then, for every list S in the output of PFDA applied to V with the distance $d(\cdot, \cdot)$, there exists a root node $y_r \in V \cup L$ such that $S = \mathrm{de}_{\vec{F}_\ell}(y_r) \cap V$.*

Proof. The proof is in Appendix B.3. □

The inverse implication of Theorem 3.5 is not true in general unless we have the assumption of structural identifiability, again irrespective of the ordering on V [3].

Theorem 3.6. *Let $\vec{F}_\ell = (V, L, \vec{E})$ be the associated graph of a latent LDPF and define an arbitrary ordering on $V = \{y_1, ..., y_n\}$. Let $d\left(y_i, y_j\right)$ be a distance defined on V such that $d\left(y_i, y_j\right) < \infty$ if and only if y_i and y_j are related. If $\vec{F}_\ell$ is structurally identifiable, then PFDA applied to V with the distance $d(\cdot, \cdot)$ outputs the sets $\mathrm{de}_{\vec{F}_\ell}(y_r) \cap V$ for all distinct root nodes $y_r \in V \cup L$.*

Proof. The proof is in Appendix B.4. □

Observe that from Proposition 2.34, we know that the log-coherence distance $d_L\left(y_i, y_j\right) < \infty$ if and only if y_i and y_j are related, giving the following corollary [3].

Corollary 3.7. *Let $\vec{F}_\ell = (V, L, \vec{E})$ be the associated graph of a latent LDPF. If $\vec{F}_\ell$ is structurally identifiable, then PFDA applied to V and the distances $d_L(\cdot, \cdot)$ outputs the sets $\mathrm{de}_{\vec{F}_\ell}(y_{r_i}) \cap V$ for all distinct root nodes $y_{r_i} \in V \cup L$.*

3.2.2 Step B. Learn the Structure of Each Rooted Tree

In the previous section we showed that, PFDA finds lists of nodes corresponding to the visible descendants of each root in a structurally identifiable LDPF. Since the distance in Equation (2.3) is additive along the paths of a rooted tree, next step is to apply RGA to each of these lists. RGA reconstructs each individual rooted tree correctly if it is guaranteed that every hidden node has degree greater than or equal to 3 in each rooted tree (i.e., every hidden node is RGA-detectable). However, RGA fails to identify the presence of FD-detectable hidden nodes as shown in Figure 3.2.

Nonetheless, we propose an improvement on RGA, Hidden Node Detection Algorithm (HNDA), presented in Algorithm 3, to also identify FD-detectable nodes [3].

Algorithm 3 Hidden Node Detection Algorithm

Input the distances $d(y_i, y_j)$ for $y_i, y_j \in V$ and a list of nodes $V_T \subseteq V$
Output (N_T, E_T) and the distances $d(y_i, y_j)$ for $y_i, y_j \in N_T$

1: Apply RGA to V_T and $d(y_i, y_j)$ for $y_i, y_j \in V_T$
2: Let (N_T, E_T) and $d(y_i, y_j)$ for $y_i, y_j \in N_T$ be the output of Step 1
3: **for** each edge $\{y_i, y_j\} \in E_T$ **do**
4: **if** $\exists y_\ell, y_k \in V \setminus N_T$ such that $d(y_i, y_k) = \infty$, $d(y_j, y_\ell) = \infty$, $d(y_i, y_\ell) < \infty$, $d(y_j, y_k) < \infty$ **then**
5: set $N_T := N_T \cup \{y_h\}$
6: set $E_T := \Big(E_T \setminus \{\{y_i, y_j\}\}\Big) \cup \{\{y_i, y_h\}, \{y_j, y_h\}\}$
7: set $d(y_h, y_m) := c$ for $y_m \in N_T$ where $c < \infty$ and $y_h \neq y_m$ and set $d(y_h, y_h) = 0$
8: **end if**
9: **end for**

It is straightforward to observe that the time complexity of HNDA is cubic in the number of visible nodes in the worst case scenario. The following theorem proves that HNDA is capable of correctly identifying all the hidden nodes in a structurally identifiable polyforest network [3].

Theorem 3.8. *Let $\vec{F}_\ell = (V, L, \vec{E})$ be the associated graph of a topologically and structurally identifiable LDPF. Let V_T be the set of visible nodes in a rooted tree $\vec{T}$ in $\vec{F}_\ell$. Then, HNDA applied to V_T and the distances $d_L(\cdot, \cdot)$ outputs the skeleton of $\vec{T}$.*

Proof. The proof is in Appendix B.5. □

3.2.3 Step C. Merge the Rooted Trees into the Polyforest Skeleton

After all rooted trees have been reconstructed, the next step is to merge them into a single polyforest structure. The main challenge is that a hidden node identified by HNDA in one rooted tree might be the same hidden node in another rooted tree. For example, in Figure 3.3 (Step B), nodes y_{h_2}, y_{h_4} and y_{h_6} are the same hidden node. The following proposition provides a full characterization to identify if a hidden node y_{h_i} in rooted tree $\vec{T}_i$ is the same hidden node y_{h_j} in rooted tree $\vec{T}_j$ [3].

Proposition 3.9. *Let $\mathcal{F} = (H(z), u)$ be a structurally identifiable latent LDPF with associated graph $\vec{F}_\ell = (V, L, \vec{E})$ and let $y_{h_i}, y_{h_j} \in L$ be in the rooted trees $\vec{T}_i$ and $\vec{T}_j$, respectively. We have*

$y_{h_i} = y_{h_j}$ *if and only if there exist two observable nodes* $y_u, y_w \in V$*, both present in* $\vec{T}_i$ *and* $\vec{T}_j$*, such that the unique path from* y_u *to* y_w *in* $\vec{T}_i$ *is*

$$y_u - y^{(i)}_{\pi_1} - ... - y^{(i)}_{\pi_{k-1}} - y_{h_i} - y^{(i)}_{\pi_{k+1}} - ... - y^{(i)}_{\pi_\ell} - y_w$$

and the unique path from y_u *to* y_w *in* $\vec{T}_j$ *is*

$$y_u - y^{(j)}_{\pi_1} - ... - y^{(j)}_{\pi_{k-1}} - y_{h_j} - y^{(j)}_{\pi_{k+1}} - ... - y^{(j)}_{\pi_\ell} - y_w.$$

Proof. The proof is in Appendix B.6. □

Observe that the two paths from y_u to y_w exist and are unique since $\vec{T}_i$ and $\vec{T}_j$ are trees [3].

The Polyforest Skeleton Learning Algorithm (PSLA), presented in Algorithm 4, uses the characterization of Proposition 3.9 to learn the skeleton of a structurally identifiable LDPF [3].

Algorithm 4 Polyforest Skeleton Learning Algorithm

Input the ordered set of nodes $V = \{y_1, ..., y_n\}$ and the distances $d(y_i, y_j)$ for $y_i, y_j \in V$
Output $F = (N, E)$ and the distances $d(y_i, y_j)$ for $y_i, y_j \in N$

Apply PFDA to V and $d(y_i, y_j)$, and obtain the lists of nodes S_k
for every list S_k **do**
    apply HNDA to S_k and $d(y_i, y_j)$ for $y_i, y_j \in V$, and obtain the tree $T_k = (N_k, E_k)$
end for
for every pair of distinct trees T_i and T_j **do**
    for every pair of distinct $u, w \in N_i \cap N_j \cap V$ **do**
        compute the path from u to w in T_i and define it as $p_i := (u, q_1, q_2, ..., q_m, w)$ where $q_a \in N_i$ for $a = 1, ..., m$
        compute the path from u to w in T_j and define it as $p_j := (u, \overline{q}_1, \overline{q}_2, ..., \overline{q}_n, w)$ where $\overline{q}_a \in N_j$ for $a = 1, ..., n$
        if $q_a = y_{h_k}$ and $\overline{q}_a = y_{h_\ell}$ for some a and $y_{h_k} \in N_i \setminus S_i$ and $y_{h_\ell} \in N_j \setminus S_j$ **then**
            label y_{h_k} and y_{h_ℓ} identically in N_i and N_j
            modify the edges in E_i and E_j so that they match the new labeling of y_{h_k} and y_{h_ℓ}
        end if
    end for
end for
Define $E := \cup_i E_i$ and $N := \cup_i N_i$
for every pair of nodes $\{y_i, y_j\} \in N$ **do**
    if $\nexists N_k$ such that $y_i, y_j \in N_k$, then set $d(y_i, y_j) = \infty$
end for

Observe that in the worst case scenario, PSLA will have a quartic time complexity in the number of visible nodes in the network. Output of PSLA will be a polyforest $F = (N, E)$ where all the hidden nodes, RGA-detectable and FD-detectable, have been identified and the set of obtained edges is undirected (i.e., the output of PSLA is the skeleton of the polyforest where all the hidden nodes have been identified). This result is proven in the following theorem. Furthermore, we can prove that the distances computed by PSLA distinguish between the pair of nodes with finite and infinite distances [3].

Theorem 3.10. *Let $\mathcal{F} = (H(z), u)$ be a topologically and structurally identifiable latent LDPF with associated graph $\vec{F}_\ell = (V, L, \vec{E})$. PSLA applied to V and distances $d_L\left(y_i, y_j\right)$ for $y_i, y_j \in V$, outputs the skeleton of $\vec{F}_\ell$ and identifies pairs of nodes in the skeleton of $\vec{F}_\ell$ that have infinite distance to each other.*

Proof. The proof is in Appendix B.7. □

3.2.4 Step D. Identify the Link Orientations

As explained in previous steps, PSLA outputs the skeleton of the polyforest from the knowledge of the log-coherence distance. In general, some a priori knowledge about the orientations might be available (e.g., certain edges could physically admit orientations only in one direction). We propose a result that can be used to determine the direction of links in an LDPF by extracting available features from the data or exploiting some a priori knowledge. The following lemma infers the direction of two edges $\{y_i, y_k\}$ and $\{y_k, y_j\}$ in the identified skeleton if $d_L\left(y_i, y_j\right) = \infty$.

Lemma 3.11. *Let $\vec{F} = (N, \vec{E})$ be a polyforest and let $y_i, y_j, y_k \in N$. If y_k is the only node on the path from y_i to y_j, and $d_L\left(y_i, y_j\right) = \infty$, then the link orientation on this path can be fully identified as $y_i \to y_k \leftarrow y_j$.*

Proof. The proof is in Appendix B.8 □

If, instead, the orientation of the edge (y_i, y_k) is known a priori, the following lemma infers the orientation of all edges of the form $\{y_k, y_j\}$ in the identified skeleton.

Lemma 3.12. *Let $\vec{F} = (N, \vec{E})$ be a polyforest and let $y_i, y_j, y_k \in N$. If y_k is the only node on the path between y_i and y_j and also $(y_i, y_k) \in \vec{E}$, we have that $d_L\left(y_i, y_j\right) < \infty$ if and only if $(y_k, y_j) \in \vec{E}$.*

Proof. The proof is in Appendix B.9 □

Using these two lemmas, we introduce the Link Orientation Identification Algorithm (LOIA), presented in Algorithm 5, to orient the links in the skeleton of a polyforest.

Algorithm 5 Link Orientation Identification Algorithm

Input a partially directed polyforest $\bar{F} = (N, E, \vec{E})$ and the distances $d(y_i, y_j)$ for $y_i, y_j \in N$
Output the partially directed polyforest $\bar{F} = (N, E, \vec{E})$

Set $\vec{E}_t := \vec{E}$
for each $(y_i, y_j) \in \vec{E}_t$ **do**
  for each $\{y_j, y_k\} \in E$ **do**
    set $E := E \setminus \{\{y_j, y_k\}\}$
    if $d(y_i, y_k) = \infty$ **then**
      set $\vec{E} := \vec{E} \cup \{(y_k, y_j)\}$
    end if
    if $d(y_i, y_k) < \infty$ **then**
      set $(N_o, E_o, \vec{E}_o)$ as the output of LOIA applied to $(N, E, \{(y_j, y_k)\})$ and $d(\cdot, \cdot)$
      set $\vec{E} := \vec{E} \cup \{(y_j, y_k)\} \cup \vec{E}_o$ and $E := E_o$
    end if
  end for
end for

Every time LOIA is called recursively, it either orients one additional edge or it exits. Since the maximum number of edges that can be oriented is linear in the number of nodes for a tree, we conclude that the time complexity of LOIA is linear in the worst case scenario.

Now we introduce the following theorem to show which edges are oriented after applying LOIA.

Theorem 3.13. *Let $\vec{F} = (N, \vec{E})$ be the associated graph of an LDPF and let $\bar{F} = (N, E', \vec{E}')$ be a partially directed polyforest with the same skeleton as $\vec{F}$ and $\vec{E}' \subseteq \vec{E}$. Let $d_L\left(y_i, y_j\right)$ be the distances for $y_i, y_j \in N$. If $(y_i, y_j) \in \vec{E}'$, then all edges $\{y_k, y_\ell\} \in E'$ for $y_k, y_\ell \in \mathrm{de}_{\vec{F}}\left(y_j\right) \cup \mathrm{pa}_{\vec{F}}\left(\mathrm{de}_{\vec{F}}\left(y_j\right)\right)$ will be oriented by LOIA applied to $\bar{F}$ and $d_L\left(y_i, y_j\right)$.*

Proof. The proof is in Appendix B.10. □

Using Theorem 3.13, if the orientation of some links is known a priori or if Lemma 3.11 can be applied, then we can initialize $\vec{E}$ with these edges and we can propagate the direction of links involving the nodes in $\mathrm{de}_{\vec{F}}\left(y_j\right) \cup \mathrm{pa}_{\vec{F}}\left(\mathrm{de}_{\vec{F}}\left(y_j\right)\right)$. Note that LOIA is a generalization of the

Generating Polytree (GPT) recovery algorithm presented in [68] because if there is some a priori information about the link orientations such as knowledge about the strict causality of the transfer functions, LOIA is able to find the direction of all edges mentioned in Theorem 3.13.

3.2.5 Putting It All Together

In this section, we present all the results developed in previous sections and present the Polyforest Learning Algorithm (PLA) in Algorithm 6.

Algorithm 6 Polyforest Learning Algorithm

Input a set of nodes V and the distances $d(y_i, y_j)$ for $y_i, y_j \in V$
Output the partially directed polyforest $\bar{F} = (N, E, \vec{E})$

1: Set $F = (N, E)$ to the output of PSLA applied to V and $d(\cdot, \cdot)$
2: Initialize $\vec{E}$ with any a priori knowledge about the link orientations and remove the corresponding edges from E
3: Set $\bar{F} = (N, E, \vec{E})$ to the output of LOIA applied to $\bar{F} = (N, E, \vec{E})$ and $d(\cdot, \cdot)$

Note that we can apply PLA to any type of network so long as we can find an additive metric along the paths of each rooted tree. For example, in [53] two metrics are provided that have the property of being additive along the paths of rooted trees. The following metric is used in the case of Guassian random variables

$$d(y_i, y_j) = -\log \left|\rho_{ij}\right| \tag{3.1}$$

where ρ_{ij} is the correlation coefficient between two random variables y_i and y_j, and the following metric is used in the case of discrete random variables

$$d(y_i, y_j) = -\log \frac{\left|\det J^{ij}\right|}{\sqrt{\det M^i \det M^j}} \tag{3.2}$$

where J^{ij} is the joint probability matrix between y_i and y_j, and M^i is the diagonal marginal probability matrix of y_i.

3.3 Numerical Example

In this section we provide an example to demonstrate how our proposed PSLA algorithm performs when applied to data in order to learn the skeleton of a network. We consider an LDPF with associated graph of Figure 3.4. Observe that the hidden nodes constitute a significant fraction of the total number of nodes making the learning process relatively challenging. These unobservable nodes (i.e., $y_{h_1} := y_{11}$, $y_{h_2} := y_{12}$, $y_{h_3} := y_{13}$ and $y_{h_4} := y_{14}$) are illustrated with dotted lines.

In our simulations, we randomly select transfer functions for the links with the following form

$$H_{ij}(z) = c_0 + c_1 z^{-1} + c_2 z^{-2} + c_3 z^{-3} \tag{3.3}$$

where $c_0 = 1$, $c_i = a_i c_{i-1}$ for $i = 1, 2, 3$ and a_i are independent random variables uniformly distributed in $\left[-\frac{1}{2}, \frac{1}{2}\right]$. For this network, we generate time series of different lengths using independent identically distributed Gaussian random processes for the inputs u_i where $i = 1, ..., 14$. The variance of u_i is set to 10% of the variance of y_i. For each time series length, we run 5000 Monte Carlo simulations and compute the log-coherence distance using the off-the-shelf *mscohere* function of MATLAB which implements the Fast Fourier Transform (FFT) via Welch's overlapping window method. The window size is chosen as the closest power of 2 to 10% of the time series length [3].

We apply PSLA to the simulated data and as discussed in previous sections, PSLA makes use of RGA. One step in RGA is the Sibling Grouping test (see Lemma 4 in [53]) which tests, for a pair of nodes y_i and y_j, if $|d_{ik} - d_{jk}|$ is equal to d_{ij} or less than d_{ij}, where $d_{ab} := d_L(y_a, y_b)$. Since the distances are estimated from data, the equality in this test is unlikely to be exactly verified. Therefore, we

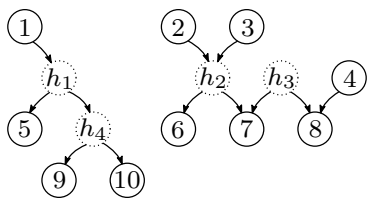


Figure 3.4: Associated graph of a LDPF [3]

implement a more robust test checking if $|d_{ik} - d_{jk}| \geq (1-\epsilon)\, d_{ij}$ or if $|d_{ik} - d_{jk}| < (1-\epsilon)\, d_{ij}$, where ϵ is a relative tolerance. By applying this robust test for $\epsilon = 20\%$ and providing PSLA with only the observations of the visible nodes, we obtain the solid curved line of Figure 3.5 in which the error bars represent a 99.99% confidence interval computed using the Wilson score [78]. This figure shows that the success rate for detecting edges approaches to 1 when the length of the time series goes to infinity confirming the theoretical results of this dissertation. Also, Figure 3.6 shows the probability of detecting a wrong link in logarithmic scale [3].

As an additional comparison, we provide our implementation of PSLA with the measurements of all the nodes, including the hidden ones, to test if it realizes that there are no actual hidden nodes in the network. The results of this experiment are plotted by the dashed curves in Figures 3.5 and 3.6. Again, when the number of samples approaches infinity, PSLA asymptotically learns the exact skeleton in accordance with our theoretical results [3].

In general, we expect to achieve better performance in the case where the information about all the network nodes is provided to PSLA as opposed to the case where only the information of a subset of nodes is provided. This is due to the fact that in the latter case, PSLA would have to detect the presence of unobserved nodes. However, in Figure 3.5, this happens only for longer time series. The explanation of this phenomenon is related to the tolerance ϵ in our implementation. In the case of longer time series, the distances are computed more accurately. Therefore, ϵ plays a minor role since it can be made arbitrarily small, still obtaining correct results. In the case of

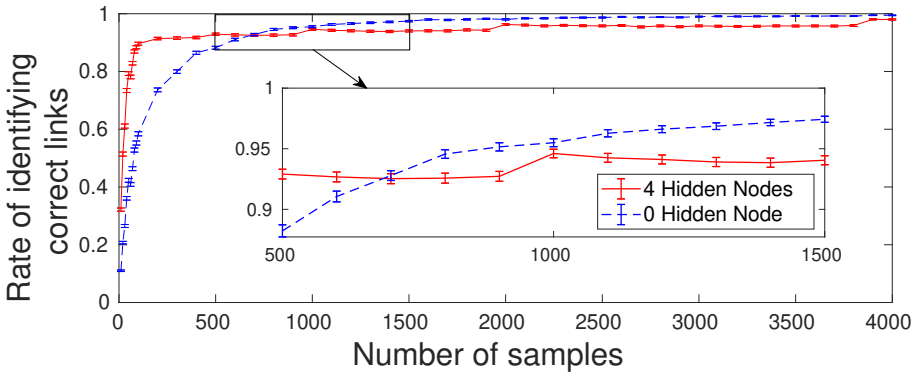


Figure 3.5: Number of samples vs. success rate in reconstruction of the graph of Figure 3.4 via PSLA [3].

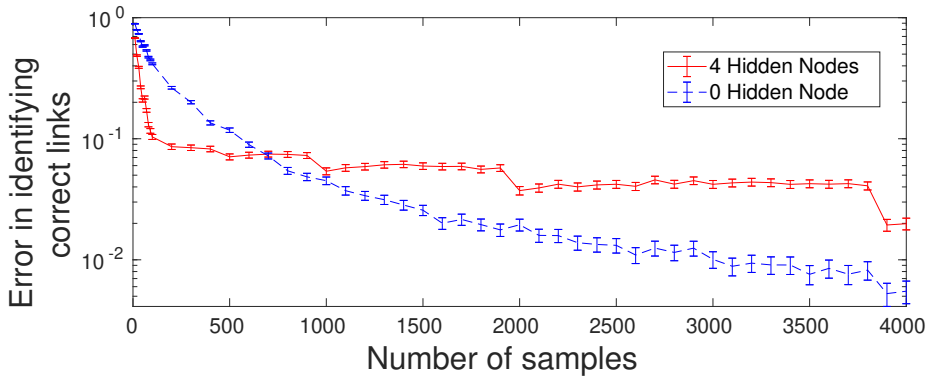


Figure 3.6: Number of samples vs. probability of detecting a wrong link in logarithmic scale for the graph of Figure 3.4 via PSLA [3].

shorter time series, the distances are less accurate, requiring a larger tolerance to obtain the correct results. In general, smaller values of ϵ lead to detecting a larger number of hidden nodes (either correctly or incorrectly). This results in an artifact in the accuracy of the reconstruction for short time series as in Figure 3.5, where the dashed curve has counterintuitively a worse performance than the solid curve. Indeed, such an artifact can be explained in the following way. Since we use the same value of ϵ for all lengths of the time series, for short time series the tolerance is smaller than its optimal value, pushing PSLA to detect more hidden nodes. Thus, in the case where the information about all the nodes is provided to PSLA, the algorithm is still pushed to detect hidden nodes (even though there are none), severely deteriorating its performance [3].

3.4 Fundamental Limitations

In this section, we show that PSLA achieves fundamental limitations for learning an LDPF. A latent LDPF $\vec{F}_\ell$ is minimal if there is no other latent LDPF with the same processes as observable nodes and fewer hidden nodes. Thus, if $\vec{F}_\ell$ is not minimal, then there is another latent LDPF $\vec{F}_\ell^{(1)}$ with the same observable nodes and fewer hidden nodes. If $\vec{F}_\ell^{(1)}$ is, again, not minimal, then there must exist $\vec{F}_\ell^{(2)}$ with the same observable nodes and even fewer hidden nodes. By iterating this statement, it is straightforward to conclude that for every latent LDPF $\vec{F}_\ell$, there is always a minimal one, $\vec{F}_\ell^{(min)}$, with the same observable nodes [3].

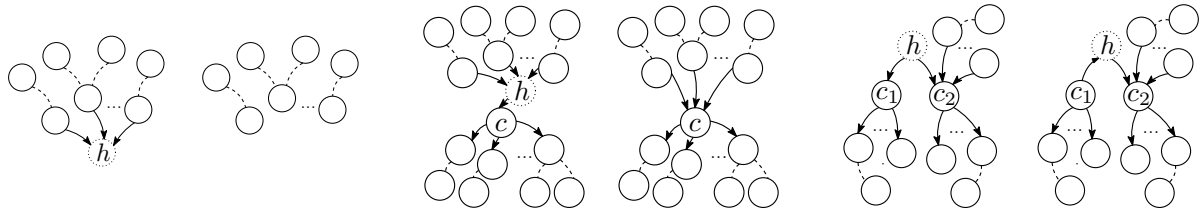
Theorem 3.14 proves that if a latent LDPF is not structurally identifiable, then the LDPF is not minimal. In other words, if a latent LDPF is minimal, it is necessarily structurally identifiable. Thus, previous sections have already shown that every minimal latent LDPF can be consistently reconstructed by PSLA. Instead, if $\vec{F}_\ell$ is not minimal, then there exists a minimal latent LDPF $\vec{F}_\ell^{(min)}$ with the same observable nodes. Since only the outputs of the observable nodes are accessible, there is no procedure capable of distinguishing between $\vec{F}_\ell$ and $\vec{F}_\ell^{(min)}$. Then, PSLA applied to the observable outputs of $\vec{F}_\ell$ necessarily reconstructs the skeleton of $\vec{F}_\ell^{(min)}$. This also shows that the skeleton of all minimal latent LDPFs with the same observable nodes of $\vec{F}_\ell$ are identical. Theorem 3.14, which enables all of these conclusions, can now be formally proven [3].

Theorem 3.14. *If a topologically identifiable latent LDPF $\mathcal{F} = (H(z), u)$ with the associated graph $\vec{F}_\ell = (V, L, \vec{E})$ is not structurally identifiable, then there is another topologically identifiable latent LDPF $\mathcal{F}' = (H'(z), \epsilon)$ with the associated graph $\vec{F}_\ell' = (V', L', \vec{E}')$ such that $V' = V$ and $L' \subset L$.*

Proof. If $\vec{F}_\ell = (V, L, \vec{E})$ is not structurally identifiable, then there is a latent node y_h that does not satisfy the degree conditions of structural identifiability. Thus, we necessarily have $\deg^+_{\vec{F}_\ell}(y_h) < 3$. Therefore, we distinguish the following three cases.

1. Case I - $\deg^+_F(y_h) = 0$: Let $N = V \cup L = \{y_1, y_2, ..., y_n\}$ be the set of all vertices in $\vec{F}_\ell$ as illustrated in Figure 3.7 (a). Dynamics of $\mathcal{F}$ has the form

$$y_i = \sum_{y_{p_i} \in \text{pa}_{\vec{F}_\ell}(y_i)} H_{ip_i}(z)\, y_{p_i} + u_i, \qquad y_h = \sum_{y_{p_h} \in \text{pa}_{\vec{F}_\ell}(y_h)} H_{hp_h}(z)\, y_{p_h} + u_h, \tag{3.4}$$



(a) (b) (c) (d) (e) (f)

Figure 3.7: Case I - $\deg^+_{\vec{F}_\ell}(y_h) = 0$: associated graph of latent LDPFs $\mathcal{F}$ (a) and $\mathcal{F}'$ (b), Case II - $\deg^+_{\vec{F}_\ell}(y_h) = 1$: associated graph of latent LDPFs $\mathcal{F}$ (c) and $\mathcal{F}'$ (d), and Case III - $\deg^+_{\vec{F}_\ell}(y_h) = 2$ and $\deg^-_{\vec{F}_\ell}(y_{c_1}) = 1$: associated graph of latent LDPFs $\mathcal{F}$ (e) and $\mathcal{F}'$ (f) [3].

for $y_i \in N_{h^-} := N \setminus \{y_h\}$. Now define a new latent LDPF $\mathcal{F}' = (H'(z), \epsilon)$ system where

$$x_i := y_i, \qquad \epsilon_i := u_i, \qquad H'_{ij}(z) := H_{ij}(z), \tag{3.5}$$

for all $y_i, y_j \in N_{h^-}$ and $y_i \neq y_j$ as illustrated in Figure 3.7 (b). Since $\deg^+_{\vec{F}_\ell}(y_h) = 0$ implies that y_h is not a parent of any y_i such that $y_i \in N_{h^-}$, the processes x_i satisfy

$$x_i = \sum_{x_{p_i} \in \mathrm{pa}_{\vec{F}'_\ell}(x_i)} H'_{ip_i}(z)\, x_{p_i} + \epsilon_i. \tag{3.6}$$

Observe that $\epsilon_i \perp\!\!\!\perp \epsilon_j$ for all $y_i, y_j \in N_{h^-}$ with $y_i \neq y_j$. Therefore, the associated graph of the latent LDPF $\mathcal{F}'$, namely $\vec{F}'_\ell = (V, L \setminus \{y_h\}, \vec{E} \setminus \{(y_i, y_h)\})$ with $y_i \in \mathrm{pa}_{\vec{F}_\ell}(y_h)$, is a latent polyforest which has the same observable nodes as $\vec{F}_\ell$ but one fewer hidden node.

2. Case II - $\deg^+_{\vec{F}_\ell}(y_h) = 1$: Let $N = V \cup L = \{y_1, y_2, ..., y_n\}$ be the set of all vertices in $\vec{F}_\ell$ as illustrated in Figure 3.7 (c). Dynamics of $\mathcal{F}$ has the form

$$y_i = \sum_{y_{p_i} \in \mathrm{pa}_{\vec{F}_\ell}(y_i)} H_{ip_i}(z)\, y_{p_i} + u_i, \qquad y_h = \sum_{y_{p_h} \in \mathrm{pa}_{\vec{F}_\ell}(y_h)} H_{hp_h}(z)\, y_{p_h} + u_h, \tag{3.7}$$

for $y_i \in N_{h^-} := N \setminus \{y_h\}$. Since $\deg^+_{\vec{F}_\ell}(y_h) = 1$, let y_c be the unique child of y_h. The process y_c satisfies the following equation

$$y_c = H_{ch}(z)\, y_h + u_c, \tag{3.8}$$

and using Equation (3.7) we have

$$y_c = \sum_{y_{p_h} \in \mathrm{pa}_{\vec{F}_\ell}(y_h)} H_{ch}(z) H_{hp_h}(z)\, y_{p_h} + H_{ch}(z) u_h + u_c. \tag{3.9}$$

Now define a new latent LDPF $\mathcal{F}' = (H'(z), \epsilon)$ system, as in Figure 3.7 (d), where

$$\begin{array}{lll} x_i := y_i, & \epsilon_i := u_i, & \epsilon_c := H_{ch}(z)u_h + u_c, \\ x_c := \sum_{x_p \in \mathrm{pa}_{\vec{F}_\ell}(y_h)\}} H'_{cp}(z) x_p + \epsilon_c, & H'_{cp}(z) := H_{ch}(z) H_{hp}(z), & H'_{ij}(z) := H_{ij}(z), \end{array} \tag{3.10}$$

for all $y_i \in N_{h^-} \setminus \{y_c\}$ and $\forall y_j \in N_{h^-}$ where $y_i \neq y_j$. Observe that $\epsilon_i \perp\!\!\!\perp \epsilon_j$ for all $y_i, y_j \in N_{h^-} \setminus \{y_c\}$ and $y_i \neq y_j$. Also $\epsilon_c \perp\!\!\!\perp \epsilon_i$ for all $y_i \in N_{h^-} \setminus \{y_c\}$. Therefore, associated graph of the latent LDPF $\mathcal{F}'$, namely $\vec{F}'_\ell = (V, L \setminus \{y_h\}, \vec{E} \cup A)$ with $y_i \in \mathrm{pa}_{\vec{F}_\ell}(y_h)$ and $A = \{(y_i, y_c)\} \setminus \{(y_i, y_h)\}$, is a latent polyforest which has the same observable nodes as $\vec{F}_\ell$ but one fewer hidden node.

3. Case III - $\deg^+_{\vec{F}_\ell}(y_h) = 2$: Since y_h does not satisfy structural identifiability conditions, we have that $\deg^-_{\vec{F}_\ell}(y_h) = 0$. Also, if $\{y_{c_1}, y_{c_2}\} = \mathrm{ch}_{\vec{F}_\ell}(y_h)$, then we have $\deg^-_{\vec{F}_\ell}(y_{c_1}) = 1$ or $\deg^-_{\vec{F}_\ell}(y_{c_2}) = 1$. With no loss of generality, consider $\deg^-_{\vec{F}_\ell}(y_{c_1}) = 1$ as in Figure 3.7 (e). Let $N = V \cup L = \{y_1, y_2, ..., y_n\}$ be the set of all vertices in $\vec{F}_\ell$. Dynamics of $\mathcal{F}$ has the form

$$\begin{aligned} y_h &= u_h, & y_i &= \textstyle\sum_{y_{p_i} \in \mathrm{pa}_{\vec{F}_\ell}(y_i)} H_{ip_i}(z)\, y_{p_i} + u_i, \\ y_{c_1} &= H_{c_1 h}(z) y_h + u_{c_1}, & y_{c_2} &= \textstyle\sum_{y_{p_i} \in \mathrm{pa}_{\vec{F}_\ell}(y_{c_2}) \setminus \{y_h\}} H_{c_2 p_i}(z)\, y_{p_i} + H_{c_2 h}(z) y_h + u_{c_2}, \end{aligned} \tag{3.11}$$

for all $y_i \in N^- := N \setminus \{y_h, y_{c_1}, y_{c_2}\}$. Now define a new latent LDPF $\mathcal{F}' = (H(z)', \epsilon)$ system, as in Figure 3.7 (f), where

$$\begin{aligned} x_i &:= y_i, & \epsilon_i &:= u_i, & H'_{jk}(z) &:= H_{jk}(z), & H'_{hc_1}(z) &:= W_{hc_1}, \\ x_h &:= H'_{hc_1}(z) x_{c_1} + \epsilon_h, & \epsilon_h &:= u_h - W_{hc_1} x_{c_1}, & x_{c_1} &:= \epsilon_{c_1}, \\ \epsilon_{c_1} &:= H_{c_1 h}(z) u_h + u_{c_1}, & x_{c_2} &:= y_{c_2}, & \epsilon_{c_2} &:= u_{c_2}, \end{aligned} \tag{3.12}$$

for all $y_i \in N^-$, $y_j \in N \setminus \{y_h\}$, $y_k \in N$, and W_{hc_1} is the Wiener filter estimating y_h using y_{c_1}. Observe that $\epsilon_i \perp\!\!\!\perp \epsilon_j$ for all $y_i, y_j \in N^-$ and $y_i \neq y_j$. Also $\epsilon_{c_1} \perp\!\!\!\perp \epsilon_{c_2}$, $\epsilon_{c_1} \perp\!\!\!\perp \epsilon_i$ and $\epsilon_{c_2} \perp\!\!\!\perp \epsilon_i$ for all $y_i \in N^-$. Using the property of the Wiener filter, we know that $\epsilon_h \perp\!\!\!\perp x_{c_1}$ which implies that $\epsilon_h \perp\!\!\!\perp \epsilon_{c_1}$. Since $u_h \perp\!\!\!\perp u_{c_2}$ and $u_{c_1} \perp\!\!\!\perp u_{c_2}$, we have that $\epsilon_h \perp\!\!\!\perp \epsilon_{c_2}$. Additionally, we know that $u_h \perp\!\!\!\perp u_i$ and $u_{c_1} \perp\!\!\!\perp u_i$ where $y_i \in N^-$ which implies that $\epsilon_h \perp\!\!\!\perp \epsilon_i$. Notice that the output processes of $\vec{F}'_\ell$ are the same as the output processes of $\vec{F}_\ell$ and so is its skeleton. However, the node y_h in $\vec{F}'_\ell$ has outdegree 1. Thus, we follow the approach in Case II and from $\vec{F}'_\ell$ we can find a latent LDPF $\mathcal{F}''$, namely $\vec{F}''_\ell = (V, L \setminus \{y_h\}, \vec{E} \cup \{(y_{c_1}, y_{c_2})\} \setminus \{(y_h, y_{c_1}), (y_h, y_{c_2})\})$ which is a latent polyforest that has the same observable nodes as $\vec{F}_\ell$ but one fewer hidden node.

□

Theorems 3.10 and 3.14 together provide the necessary and sufficient conditions for reconstructing an LDPF. Indeed, Theorem 3.10 shows that if a polyforest is structurally identifiable, then it is possible to learn its skeleton from the knowledge of the distances of the visible nodes (sufficiency). Theorem 3.14 shows that if a polyforest is not structurally identifiable, then it is not possible to reconstruct its skeleton from the knowledge of the distances of the visible nodes since there exists at least one other latent polyforest with the same visible nodes but fewer number of hidden nodes (necessity) [3].

Chapter 4

Learning Non-linear Networks with Tree Structures

In this chapter, we develop an algorithm for learning polytree networks for generic distributions considering the presence of hidden nodes. Furthermore, as opposed to the results of previous chapter, the assumption of linearity of the network is not exploited in this method but the statistics of the observed data up to the third order are required. We also provide the fundamental limitations of solving the problem of learning polytree networks under these assumptions.

It is worth to mention that the methods in this chapter are developed for polytree networks, however, we can apply the same algorithms to learn the structure of a polyforest. This extension is simply possible since the proposed procedure inherently splits the rooted subtrees in the network and then tries to learn the structure of the original polytree (or polyforest) at the same time as recovering the structure of each rooted subtree.

4.1 An Algorithm to Learn Latent Polytree Networks

As mentioned before, we assume that we only have access to the measurements of the observed variables and the goal is to recover the network structure (including the hidden variables). By observing the visible nodes (or variables), we can extract independence statements of the form $\mathcal{I}(y_i, \emptyset, y_j)$ or $\neg\mathcal{I}(y_i, \emptyset, y_j)$ (namely, the second order statistics) and statements of the form

$\mathcal{I}(y_i, y_k, y_j)$ or $\neg\mathcal{I}(y_i, y_k, y_j)$ (namely, the third order statistics) where y_i, y_j and y_k are the observed variables.

Here, we propose an algorithm that takes the second and third order statistics of the observed nodes in a minimal polytree network and learns its pattern. This algorithm consists of 5 tasks [4]:

1. From the independence statements involving the visible nodes, determine the number of rooted subtrees in the latent polytree and their respective sets of visible nodes;

2. Given all the visible nodes belonging to each rooted subtree, determine the collapsed quasi-skeleton of each rooted subtree;

3. Merge the hidden clusters in the collapsed quasi-skeleton of each rooted subtree given that they partially overlap to obtain the collapsed quasi-skeleton of the latent polytree;

4. Determine the quasi-skeleton of the latent polytree from the collapsed quasi-skeleton of the latent polytree (recover Type-I hidden nodes);

5. Obtain the pattern of the latent polytree from the quasi-skeleton of the latent polytree (recover some edge orientations and all Type-II hidden nodes).

Consider a minimal latent polytree as depicted in Figure 4.1 (True). A step by step output of the proposed algorithm is depicted in Figures 4.1 (Task 1) - (Task 5). Observe that the full polytree is not recovered at the end of Task 5 since one edge is left undirected but the pattern of the polytree is learned. The following subsections provide the details of each step and the technical results developed to support this algorithm. We stress that the first task is basically leveraging the PFDA algorithm developed in Subsection 3.2.1 and [3], and the second task leverages the results developed in [79] for recovering the structure of rooted trees. The main novel results in this chapter lie in Tasks 3-5 of the algorithm [4].

4.1.1 Task 1. Determine the Visible Nodes of Each Rooted Subtree

This first task can be performed by the PFDA, presented in Subsection 3.2.1 and [3]. The main purpose of the PFDA is to recover the lists of all visible nodes in each rooted tree of a minimal latent polytree denoted by $\vec{P}_\ell = (V, L, \vec{E})$ [4]. As explained in Subsection 3.2.1, PFDA takes

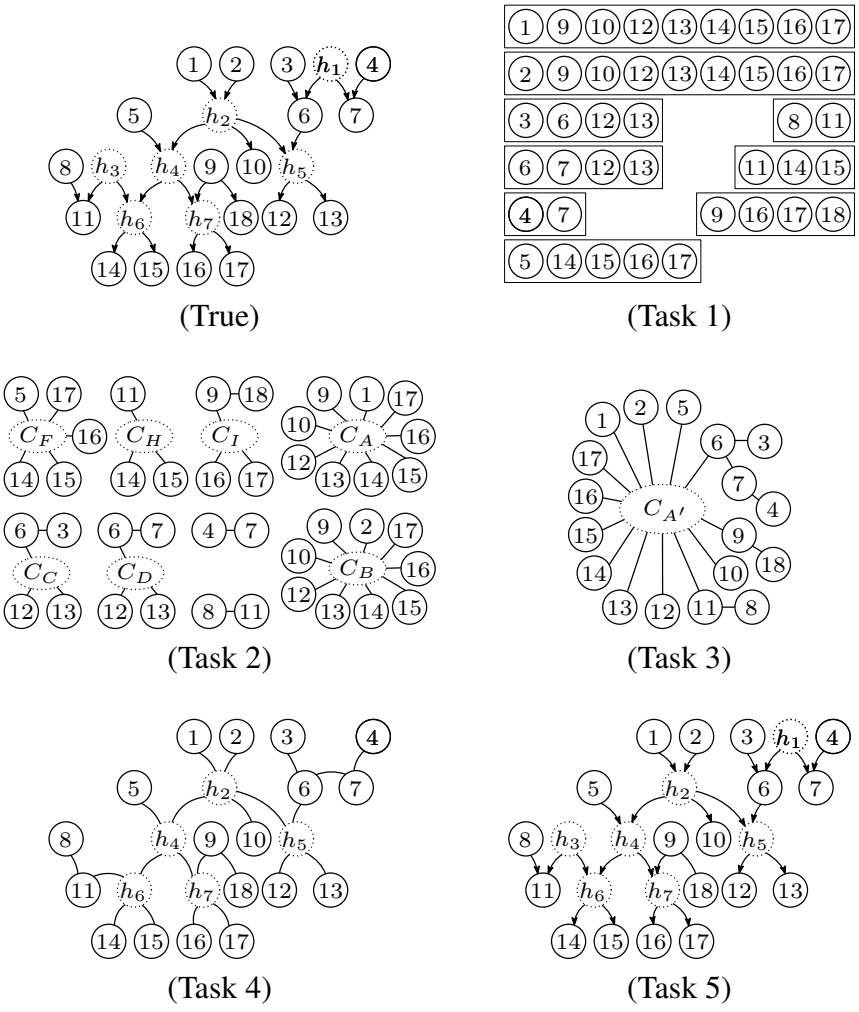


Figure 4.1: The actual minimal latent polytree (True), the lists of visible nodes for each rooted subtree (Task 1), collapsed quasi-skeletons of each rooted subtree (Task 2), merging of the overlapping hidden clusters (Task 3), detection of Type-I hidden nodes (Task 4), and detection of Type-II hidden nodes along with orientation of the edges to obtain the pattern (Task 5). Observe that the full polytree is not recovered at the end of Task 5 since the edge $y_9 - y_{18}$ is left undirected but the pattern of the polytree is learned [4].

as input the set of visible nodes of $\vec{P}_\ell$ and a metric d with the property that $d(y_i, y_j) < \infty$ if and only if y_i, y_j are in the same rooted subtree. In this case, PFDA is proven to output sets of visible nodes with the property that each set corresponds to the visible descendants of a root of $\vec{P}_\ell$. However, here we would like to achieve the same output with a slightly different type of input which is the independence relations of the form $\mathcal{I}(y_i, \emptyset, y_j)$ or $\neg\mathcal{I}(y_i, \emptyset, y_j)$. Since we have $\neg\mathcal{I}(y_i, \emptyset, y_j)$ if and only if y_i, y_j are in the same rooted subtree, it is immediate to verify that the relations $\mathcal{I}(y_i, \emptyset, y_j)$ or $\neg\mathcal{I}(y_i, \emptyset, y_j)$ can replace the role of the additive metric in the algorithm. This is precisely what we need for implementing Task 1. We report an equivalent version of PFDA, presented in Algorithm 7 for completeness, which takes as input the independence relations of the form $\mathcal{I}(y_i, \emptyset, y_j)$ or $\neg\mathcal{I}(y_i, \emptyset, y_j)$ instead of an additive metric [4].

Algorithm 7 Pairwise-Finite Distance Algorithm

Input the ordered set of nodes $V = \{y_1, ..., y_n\}$ and the statements of the form $\mathcal{I}(y_i, \emptyset, y_j)$ or $\neg\mathcal{I}(y_i, \emptyset, y_j)$ for $y_i, y_j \in V$

Output the set of all non-eliminated lists $S_{i,j}$

for every node $y_i \in V$ such that $\forall y_j \in V \setminus \{y_i\}$ we have that $\mathcal{I}(y_i, \emptyset, y_j)$ **do**
    define $S_{i,0} := \{y_i\}$
end for
for each pair $y_i, y_j \in V$ with $i < j$, and $\neg\mathcal{I}(y_i, \emptyset, y_j)$ **do**
    define $S_{i,j} := \{y_i, y_j\}$
    for each $y_k \in V \setminus S_{i,j}$ **do**
        if $\forall y \in S_{i,j} : \neg\mathcal{I}(y_k, \emptyset, y)$, then add y_k to $S_{i,j}$
    end for
end for
for each pair $y_i, y_j \in V$ with $i < j$ **do**
    if $S_{i,j} = S_{k,\ell}$ for some k and ℓ, then eliminate $S_{k,\ell}$
end for

The following theorem shows that the output of PFDA applied to the independence statements is the lists of visible nodes belonging to the same rooted subtree of $\vec{P}_\ell$ [4].

Theorem 4.1. *Consider a minimal latent polytree $\vec{P}_\ell = (V, L, \vec{E})$ faithful to a probabilistic model. Then PFDA applied to the independence statements $\mathcal{I}(y_i, \emptyset, y_j)$ or $\neg\mathcal{I}(y_i, \emptyset, y_j)$ of the probabilistic model for $y_i, y_j \in V$, outputs a collection of sets such that each of them is given by all the visible descendants of a root of $\vec{P}_\ell$.*

Proof. The proof is in Appendix C.1. □

4.1.2 Task 2. Determine the Collapsed Representation of the Quasi-skeleton of Each Rooted Subtree

The second task can be performed by the Reconstruction Algorithm for Latent Rooted Trees proposed in [79]. We report it as Algorithm 8 here for completeness and to match the notation of this dissertation. The input of this algorithm is the set V_r of visible nodes belonging to a rooted subtree T_r of the minimal latent polytree and also the independence relations of the form $\mathcal{I}(y_i, y_k, y_j)$ or $\neg\mathcal{I}(y_i, y_k, y_j)$ for distinct $y_i, y_j, y_k \in V_r$. Its output, then, is the collapsed quasi-skeleton of the rooted subtree T_r. For completeness, we have included the intuition and a detailed explanation of the Reconstruction Algorithm for Latent Rooted Trees in Appendix C.2 [4].

Algorithm 8 Reconstruction Algorithm for Latent Rooted Trees

Input the set of visible nodes in a rooted subtree V_r and the independence statements of the form $\mathcal{I}(y_i, y_k, y_j)$ or $\neg\mathcal{I}(y_i, y_k, y_j)$ for $y_i, y_j, y_k \in V_r$

Output (V_r, L_r, E_r) the collapsed quasi-skeleton of T_r

1: Initialize $V_{temp} := V_r$, $L_r := \{\}$, and $E_r := \{\}$
2: If $|V_{temp}| = 2$, i.e., $V_{temp} = \{y_i, y_j\}$, then add the edge $y_i - y_j$ to E_r and if $n \leq 2$, stop and output the results
3: Determine a visible terminal node y_k in the rooted tree by verifying the condition $\neg\mathcal{I}(y_i, y_k, y_j)$ for all $y_i, y_j \in V_{temp} \setminus \{y_k\}$
4: Search for a visible node $y_\ell \in V_{temp} \setminus \{y_k\}$ linked to y_k by verifying the condition $\mathcal{I}(y_k, y_\ell, y_j)$ for all $\forall y_j \in V_{temp} \setminus \{y_k, y_\ell\}$
5: **if** y_ℓ exists **then**
6: add the link $y_\ell - y_k$ to E_r, remove y_k from V_{temp}, and go to Step 2.
7: **else**
8: create a new hidden node y_h in L_r and add the link $y_k - y_h$ to E_r
9: compute the set $K \subseteq V_{temp}$ such that $y_j \in K$ implies that $\neg\mathcal{I}(y_j, y_i, y_k)$ for all $y_i \neq y_j, y_k$ where $y_i \in V_{temp} \setminus K$
10: add $y_h - y_j$ to E_r for $y_j \in K$
11: set $(V^{(j)}, L^{(j)}, E^{(j)})$ as the output of this algorithm applied to each $V^{(j)}$ defined as the union of $\{y_j\}$ and the set of nodes in V_{temp} separated from y_k by $y_j \in K$
12: set $E_r := \bigcup_{y_j \in K} E^{(j)} \cup E_r$ and $L_r := \bigcup_{y_j \in K} L^{(j)} \cup L_r$
13: **end if**

Thus, we can call this algorithm on all of the sets of visible nodes $V_1, ..., V_{n_r}$, where n_r is the number of roots, obtained from Task 1 and find the collapsed quasi-skeletons of all the rooted subtrees of the latent polytree. This result is formalized in the following theorem [4].

Theorem 4.2. *Let $\vec{P}_\ell = (V, L, \vec{E})$ be a minimal latent polytree. Consider a root y_r of $\vec{P}_\ell$ and let $V_r = V \cap \mathrm{de}_{\vec{P}_\ell}(y_r)$. The output of Reconstruction Algorithm for Latent Rooted Trees applied to V_r and the independence relations between the nodes in V_r is the collapsed quasi-skeleton of the rooted subtree with the root y_r.*

Proof. The proof is in Appendix C.3. □

4.1.3 Task 3. Merge the Hidden Clusters of the Collapsed Rooted Subtrees

By applying the Reconstruction Algorithm for Latent Rooted Trees on each set of visible nodes in the same rooted tree, we have, as an output, the collapsed quasi-skeletons of all rooted subtrees in the original hidden polytree. In the general case, some hidden clusters in the collapsed quasi-skeleton of the rooted subtrees might overlap, namely they might share some nodes. The following theorem provides a test on the sets of visible nodes of the rooted subtrees in a minimal latent polytree to determine if two hidden clusters in two distinct collapsed quasi-skeletons of two rooted subtrees belong to the same cluster in the collapsed quasi-skeleton of the polytree [4].

Theorem 4.3. *Consider a minimal latent polytree $\vec{P}_\ell$. Let C_1 and C_2 be two distinct hidden clusters in the collapsed quasi-skeletons of two rooted subtrees of $\vec{P}_\ell$. If the set of neighbors of C_1 and the set of neighbors of C_2 share at least a pair of visible nodes, i.e., $|N(C_1) \cap N(C_2)| \geq 2$, then the nodes in C_1 and C_2 belong to the same hidden cluster in the collapsed quasi-skeleton of $\vec{P}_\ell$.*

Proof. The proof is in Appendix C.4. □

This theorem is the enabling result for the Hidden Cluster Merging Algorithm (HCMA), presented in Algorithm 9, which merges all the collapsed quasi-skeletons associated with the individual rooted subtrees, obtained from Task 2, into the collapsed quasi-skeleton of the polytree. This algorithm starts with the collapsed quasi-skeleton of the rooted subtrees, then finds pairs of clusters that overlap by testing if they share at least one pair of visible neighbors (see Theorem 4.3), and then merges the overlapping pairs. This procedure is repeated until no clusters are merged anymore [4].

Algorithm 9 Hidden Cluster Merging Algorithm

Input the collapsed quasi-skeleton of the rooted subtrees $T_i = (V_i, L_i, E_i)$ for $i = 1, ..., n_r$

Output the collapsed quasi-skeleton P of the latent polytree

1: Initialize the set of clusters $\mathcal{P}$ with the hidden clusters of all T_i, i.e., $\mathcal{P} := \{\{C_1\}, \{C_2\}, ..., \{C_k\}\}$

2: **while** there are two elements $C_i, C_j \in \mathcal{P}$ such that $|N(C_i) \cap N(C_j)| \geq 2$ **do**

3: remove C_i, C_j from $\mathcal{P}$ and add $C_i \cup C_j$ to $\mathcal{P}$

4: define $N(C_i \cup C_j) := N(C_i) \cup N(C_j)$

5: **end while**

6: Define the polytree $P = (\cup_i V_i, \mathcal{P}, E)$ where $E := \{\{y_a, y_b\} \mid \exists\, i : y_a, y_b \in V_i, y_a - y_b \in E_i\} \cup \{\{y_a, C_b\} \mid \exists\, i, h : y_a \in V_i, y_h \in L_i, L_i \subseteq C_b, C_b \in \mathcal{P}, y_a - y_h \in E_i\}$

The following theorem guarantees that, for a minimal latent polytree, the output of HCMA is the collapsed quasi-skeleton of the polytree [4].

Theorem 4.4. *Let $\vec{P}_\ell = (V, L, \vec{E})$ be a minimal latent polytree and let $T_i = (V_i, L_i, E_i)$ for $i = 1, ..., n_r$ be the collapsed quasi-skeletons of the rooted subtrees of $\vec{P}_\ell$. Then HCMA outputs the collapsed quasi-skeleton of $\vec{P}_\ell$.*

Proof. The proof is in Appendix C.5. □

4.1.4 Task 4. Determine the Quasi-skeleton of the Latent Polytree from the Collapsed Quasi-skeleton of the Latent Polytree

After performing the HCMA, the output is the collapsed quasi-skeleton of the latent polytree, thus, the structure of the hidden nodes within each hidden cluster is not known yet. Note that the restriction of the original polytree to the closure of a hidden cluster is a smaller polytree. The goal of this task is to recover the structure of the hidden clusters by focusing on each individual closure (i.e., recover Type-I hidden nodes and their connectivities). Given the closure of a hidden cluster, the basic strategy is to detect one root of the hidden cluster along with the visible nodes (if any) linked to this root. Then, we label such a root as a visible node, add edges between this node and its visible neighbors, and subsequently apply the same strategy recursively to the descendants of such a detected root [4].

Since we focus on the closure of a specific hidden cluster, say C, we define the following sets $\tilde{V}_r = V_r \cap N(C)$ for $r = 1, ..., n_r$ where n_r is the number of rooted subtrees in the latent polytree and V_r are the sets of visible nodes in each rooted subtree (obtained from Task 1). A fundamental result for detection of a root of a hidden cluster is the following theorem [4].

Theorem 4.5. *Let $\vec{P}_\ell$ be a minimal latent polytree and let $\vec{T}_r = (V_r, L_r, \vec{E}_r)$ with $r = 1, ..., n_r$ be all the rooted subtrees of $\vec{P}_\ell$. Let C be a hidden cluster in the collapsed quasi-skeleton of $\vec{P}_\ell$. Define $\tilde{V}_r := V_r \cap N(C)$ for $r = 1, ..., n_r$ where n_r is the number of roots in $\vec{P}_\ell$. Then, T_r contains a hidden root of C if and only if $\tilde{V}_r \neq \emptyset$ and for all $\tilde{V}_{r'}$ with $r' \neq r$ we have $|\tilde{V}_r \setminus \tilde{V}_{r'}| > 1$ or $|\tilde{V}_{r'} \setminus \tilde{V}_r| \leq 1$.*

Proof. The proof is in Appendix C.6. □

To make the application of this theorem more clear, consider the latent polytree introduced in Figure 4.1 (True). After applying the first three tasks, we obtain the collapsed quasi-skeleton of the latent polytree as depicted in Figure 4.1 (Task 3). Observe that the rooted subtrees $\vec{T}_1$ (with root y_1) and $\vec{T}_2$ (with root y_2) satisfy the conditions of Theorem 4.5 indicating that they contain a root of the hidden cluster. The following lemma allows one to find the visible nodes linked to a hidden root in the closure of a hidden cluster [4].

Lemma 4.6. *Let $\vec{P}_\ell$ be a minimal latent polytree. Consider a hidden root y_h of a hidden cluster C in the collapsed quasi-skeleton of $\vec{P}_\ell$ where y_h belongs to the rooted subtree $T_r = (V_r, L_r, \vec{E}_r)$. Define $\tilde{V}_{r'} := V_{r'} \cap N(C)$ for $r' = 1, ..., n_r$ where n_r is the number of roots in $\vec{P}_\ell$. The visible nodes linked to y_h are given by the set $W \setminus \overline{W}$ where*

$$I := \{r\} \cup \{r' \textit{ such that } |\tilde{V}_r \setminus \tilde{V}_{r'}| = |\tilde{V}_{r'} \setminus \tilde{V}_r| = 1\}, \qquad W := \bigcup_{i \in I} \tilde{V}_i, \qquad \overline{W} := \bigcup_{i \notin I} \tilde{V}_i.$$

Proof. The proof is in Appendix C.7. □

Again, we follow the example of Figure 4.1 to show the steps of Task 4 in more details. Without loss of generality, choose $\vec{T}_r = \vec{T}_1$. Consider the closure of $C_{A'}$ obtained at the end of Task 3 and then apply Lemma 4.6 to obtain $I = \{1, 2\}$, $W = \{y_1, y_2, y_{10}, y_{12}, y_{13}, y_{14}, y_{15}, y_{16}, y_{17}\}$, and $\overline{W} = \{y_5, y_6, y_9, y_{11}, y_{12}, y_{13}, y_{14}, y_{15}, y_{16}, y_{17}\}$. Thus, we have $W \setminus \overline{W} = \{y_1, y_2, y_{10}\}$. Therefore, the visible nodes linked to the hidden root in $\vec{T}_1$ are y_1, y_2 and y_{10}.

Now we introduce the Hidden Cluster Learning Algorithm (HCLA), presented in Algorithm 10, to learn the structure of a hidden cluster [4].

Algorithm 10 Hidden Cluster Learning Algorithm

Input the collapsed quasi-skeleton of a minimal polytree $\vec{P}_\ell$, collapsed quasi-skeletons of the rooted subtrees $T_i = (V_i, L_i, E_i)$ for $i = 1, ..., n_r$, and the set of the hidden clusters $\mathcal{P} = \{C_1, ..., C_{n_C}\}$

Output P and the independence relations of the form $\mathcal{I}(y_a, \emptyset, y_b)$ or $\neg\mathcal{I}(y_a, \emptyset, y_b)$ for all nodes $y_a, y_b \in \bigcup_i V_i$

while $\mathcal{P} \neq \emptyset$ **do**
  Call Hidden Node Detection Procedure(C_1) where C_1 is the first element of $\mathcal{P}$
end while
procedure Hidden Node Detection(C)
  Compute $\tilde{V}_i = V_i \cap N(C)$
  Find $\tilde{V}_r$ which satisfies $|\tilde{V}_r \setminus \tilde{V}_{r'}| > 1$ or $|\tilde{V}_{r'} \setminus \tilde{V}_r| \leq 1$ for all $r' \neq r$ (as in Theorem 4.5)
  Initialize $W := \tilde{V}_r$, $\overline{W} := \emptyset$, and $I := \{r\}$
  for all $i = 1, ..., n_r$ with $i \neq r$ **do**
    if $|\tilde{V}_r \setminus \tilde{V}_i| = 1$ and $|\tilde{V}_i \setminus \tilde{V}_r| = 1$ (as in Lemma 4.6) **then**
      $W := W \cup \tilde{V}_i$ and $I := I \cup \{i\}$
    else
      $\overline{W} := \overline{W} \cup \tilde{V}_i$
    end if
  end for
  A new hidden node y_h is revealed
  Add y_h to all the rooted trees T_i with $i \in I$, namely $V_i := V_i \cup \{y_h\}$
  Add the independence relation $\neg\mathcal{I}(y_h, \emptyset, y)$ for all $y \in V_i$ with $i \in I$, and add the independence relation $\mathcal{I}(y_h, \emptyset, y)$ for all other nodes y
  Link all nodes in $W \setminus \overline{W}$ to y_h in all T_i with $i \in I$, namely $E_i := E_i \cup \left\{\{y_h, y\} \mid y \in W \setminus \overline{W}\right\}$
  for all $i \in I$ **do**
    create $n_k = |W \cap \overline{W}|$ new clusters: $C_1^{(i)}, ..., C_{n_k}^{(i)}$
    link y_h to $C_1^{(i)}, ..., C_{n_k}^{(i)}$
    link each cluster $C_1^{(i)}, ..., C_{n_k}^{(i)}$ to a distinct element in $W \cap \overline{W}$
  end for
  while $\exists y_a, y_b \in N(C_j^{(i)}) \cup N(C_k^{(i)})$ such that $y_a, y_b \in \tilde{V}_m$ where $m \notin I$ **do**
    merge the two hidden clusters $C_j^{(i)}$ and $C_k^{(i)}$
    update the structure of T_i with the new hidden clusters
  end while
  Let $P = (V, \mathcal{P}, E)$ be the output of HCMA applied to $T_i = (V_i, L_i, E_i)$, for $i = 1, ..., n_r$
end procedure

Again, consider the closure of the hidden cluster $C_{A'}$ as depicted in Figure 4.2 (Task 4a) which we obtained at the end of Task 3. Then, apply Hidden Node Detection procedure to $C_{A'}$ and observe that the output at the end of Step 23 of Algorithm 10 is in Figure 4.2 (Task 4b). The output of the

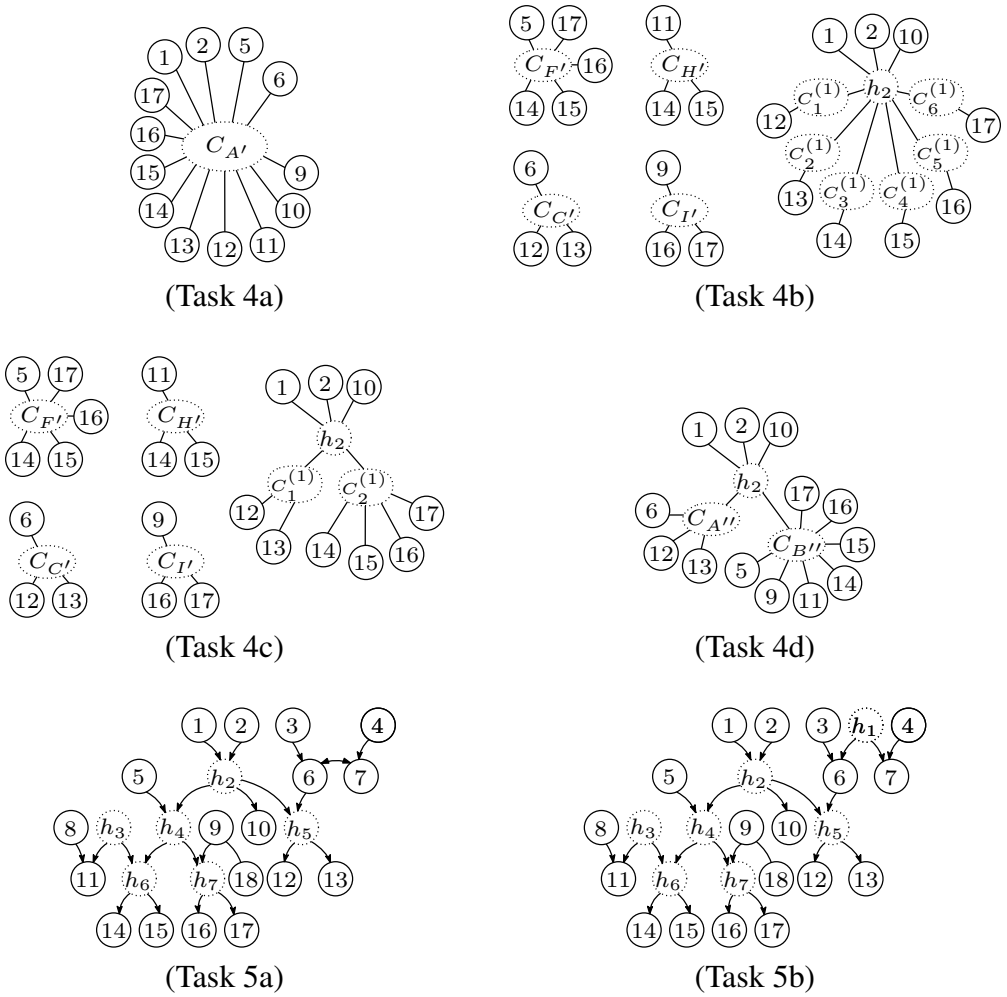


Figure 4.2: The closure of the hidden cluster $C_{A'}$ of the latent polytree in Figure 4.1(True) obtained after Task 3 (Task 4a), the hidden clusters obtained after Step 23 of HCLA (Task 4b), merging of the hidden clusters as in Steps 24-27 of HCLA (Task 4c), merging of the overlapping hidden clusters as in Step 28 of HCLA (Task 4d), orienting the edges in the quasi-skeleton of the latent polytree as in Steps 1-4 of HRRA (Task 5a), and discovering Type-II hidden nodes (Task 5b) [4].

merging in Steps 24-27 is depicted in Figure 4.2 (Task 4c) and the output of the merging in Step 28 is depicted in Figure 4.2 (Task 4d). Now, we can apply the same procedure recursively to the remaining hidden clusters to obtain the final output of Task 4, the quasi-skeleton of the polytree, as depicted in Figure 4.1 (Task 4) [4].

In the following theorem, we show that the output of HCLA is the quasi-skeleton of the latent polytree [4].

Theorem 4.7. *Let $\vec{P}_\ell = (V, L, \vec{E})$ be a minimal latent polytree. When HCLA is applied to all hidden clusters of the collapsed quasi-skeleton of $\vec{P}_\ell$, the output $P = (V, E)$ is the quasi-skeleton of $\vec{P}_\ell$. Furthermore, HCLA also outputs, for each pair $y_i, y_j \in V$, the relation $\mathcal{I}(y_i, \emptyset, y_j)$ if and only if the path connecting y_i and y_j in $\vec{P}_\ell$ contains an inverted fork.*

Proof. The proof is in Appendix C.8. □

4.1.5 Task 5. Obtain the Pattern of the Latent Polytree from the Quasi-skeleton of the Latent Polytree

Once the quasi-skeleton of the latent polytree has been obtained, the only missing nodes to recover the full skeleton are the Type-II hidden nodes of the original polytree. Interestingly, the detection of such hidden nodes can be performed concurrently with the recovery of the edge orientations. In particular, we can apply the GPT algorithm in [68] to orient the edges of the quasi-skeleton of the polytree. In this case, if any edge receives two different orientations, we show that it implies that there exists one Type-II hidden node between the two linked nodes [4].

Thus, we introduce the Hidden Root Recovery Algorithm (HRRA), presented in Algorithm 11, which is simply an implementation of the GPT algorithm (Steps 1-4), as depicted in Figure 4.2 (Task 5a), with the additional detection of Type-II hidden nodes (Steps 6-11), as depicted in Figure 4.2 (Task 5b). Observe that Steps 1-4 of HRRA implement the GPT algorithm to find as many v-structures as possible considering all the independence statements (including the ones obtained from Task 4). These steps also orient other edges so long as no new v-structure is created. After this stage, we can simply remove the edges that have been oriented from the list of unoriented edges as in Step 5. Then, in the cases where an edge has two different orientations, a new Type-II hidden node is revealed as in Steps 6-11.

Algorithm 11 Hidden Root Recovery Algorithm

Input $P = (V, E)$, the quasi-skeleton of a latent polytree, and the independence relations of the form $\mathcal{I}(y_i, \emptyset, y_j)$ or $\neg\mathcal{I}(y_i, \emptyset, y_j)$ for all nodes $y_i, y_j \in V$

Output the partially directed polytree $\bar{P} = (V, E, \vec{E})$

while additional edges are oriented **do**
    if $y_i - y_k, y_j - y_k \in E$ and $\mathcal{I}(y_i, \emptyset, y_j)$, then add $y_i \to y_k$ and $y_j \to y_k$ to $\vec{E}$
    if $y_i \to y_k \in \vec{E}$, $y_k - y_j \in E$ and $\neg\mathcal{I}(y_i, \emptyset, y_j)$, then add $y_k \to y_j$ to $\vec{E}$
end while
Remove the edges that are oriented in $\vec{E}$ from E
for all y_i, y_j such that $y_i \to y_j, y_j \to y_i \in \vec{E}$ **do**
    a new hidden node of Type-II is detected which is a parent of y_i and y_j
    remove $y_i \to y_j, y_j \to y_i$ from $\vec{E}$
    add a new node y_h to V
    add $y_h \to y_j, y_h \to y_i$ to $\vec{E}$
end for

Moreover, we provide the result stated in Theorem 4.8 to prove that HRRA outputs the pattern of the latent polytree [4].

Theorem 4.8. *Let $\vec{P}_\ell$ be a minimal latent polytree. When the input is the quasi-skeleton of $\vec{P}_\ell$ with the independence statements of the form $\mathcal{I}(y_i, \emptyset, y_j)$ or $\neg\mathcal{I}(y_i, \emptyset, y_j)$ for all the pairs of nodes y_i and y_j, the output of HRRA is the pattern of $\vec{P}_\ell$.*

Proof. The proof is in Appendix C.9. □

4.1.6 Putting It All Together

In this section, we present all the results developed in previous sections and present the Polyforest Learning Algorithm (PLA) in Algorithm 12. Note that this algorithm is called the same as the Algorithm 6 since they both learn the polyforest structure. However, they should not be confused since they exploit different assumptions and therefore their inputs are different.

Note that we can apply PLA to any type of network so long as we can find the independence statements of the form $\mathcal{I}(y_i, \emptyset, y_j)$ or $\neg\mathcal{I}(y_i, \emptyset, y_j)$, and $\mathcal{I}(y_i, y_k, y_j)$ or $\neg\mathcal{I}(y_i, y_k, y_j)$ for all the observable variables.

Algorithm 12 Polyforest Learning Algorithm

Input the ordered set of nodes $V = \{y_1, ..., y_n\}$ and the statements of the form $\mathcal{I}(y_i, \emptyset, y_j)$ or $\neg\mathcal{I}(y_i, \emptyset, y_j)$, and $\mathcal{I}(y_i, y_k, y_j)$ or $\neg\mathcal{I}(y_i, y_k, y_j)$ for $y_i, y_j, y_k \in V$

Output the pattern $\bar{P} = (N, E, \vec{E})$

1: Set S to the output of PFDA applied to V and $\mathcal{I}(y_i, \emptyset, y_j)$ or $\neg\mathcal{I}(y_i, \emptyset, y_j)$ for $y_i, y_j \in V$
2: **for** each list S_i in S **do**
3: set $T_i = (V_i, L_i, E_i)$ to the output of Reconstruction Algorithm for Latent Rooted Trees applied to S_i, and $\mathcal{I}(y_a, y_c, y_b)$ or $\neg\mathcal{I}(y_a, y_c, y_b)$ for $y_a, y_b, y_c \in S_i$
4: **end for**
5: Set $P = (V, \mathcal{P}, E)$ to the output of HCMA applied to all the T_i
6: Set P to the output of *HCLA* applied to P, all the T_i, and $\mathcal{P}$
7: Set $\bar{P} = (N, E, \vec{E})$ to the output of *HRRA* applied to P and the independence relations of the form $\mathcal{I}(y_i, \emptyset, y_j)$ or $\neg\mathcal{I}(y_i, \emptyset, y_j)$

4.2 Additional Examples

In this section we provide more examples to show how we can leverage PLA to learn the structure of different networks given that all the independence statements of the form $\mathcal{I}(y_i, \emptyset, y_j)$ or $\neg\mathcal{I}(y_i, \emptyset, y_j)$, and $\mathcal{I}(y_i, y_j, y_k)$ or $\neg\mathcal{I}(y_i, y_j, y_k)$ for all the observable nodes y_i, y_j and y_k in the network are provided.

4.2.1 A Star Network

Consider the star network depicted in Figure 4.3 (True). This network contains four rooted subtrees and one Type-I hidden node.

The output of Task 1 is the set of lists of nodes that belong to the same rooted tree as depicted in Figure 4.3 (Task 1). After applying Task 2, we get the collapsed quasi-skeletons of each rooted subtree as in Figure 4.3 (Task 2). Since all of the identified hidden clusters are connected to the pair y_5 and y_6, HCMA merges them together as in Figure 4.3 (Task 3). When we implement the tests in Theorem 4.5 and Lemma 4.6, we have $I = \{1, 2, 3, 4\}$, $W = \{y_1, y_2, y_3, y_4, y_5, y_6\}$ and $\overline{W} = \emptyset$. Thus, all the nodes are connected to the newly recovered hidden node y_{h_1}. Since there are no hidden clusters in the network anymore, namely, $\mathcal{P}$ is empty, HCLA stops and the output of Task 4 is shown in Figure 4.3 (Task 4). Note that we also recover the independence statements at the end of Task 4 which are used in HRRA in Task 5 to recover edge orientations as in Figure 4.3 (Task 5).

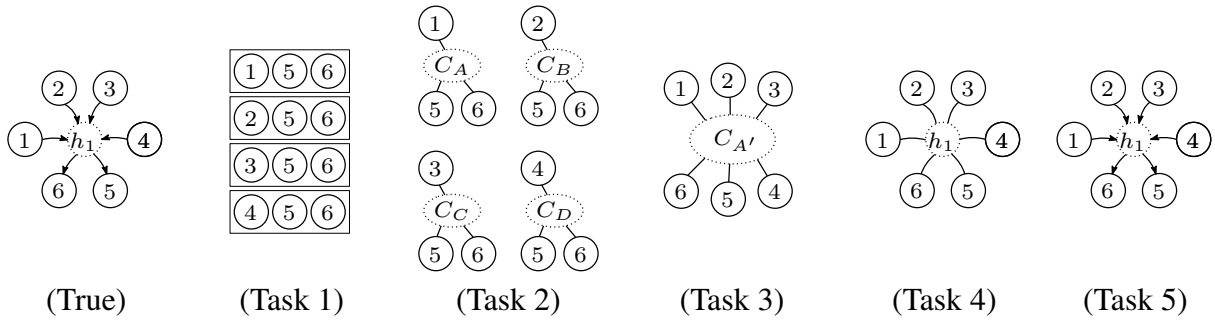


Figure 4.3: The actual minimal latent polytree (True), the lists of visible nodes for each rooted subtree (Task 1), collapsed quasi-skeletons of each rooted subtree (Task 2), merging of the overlapping hidden clusters (Task 3), detection of Type-I hidden nodes (Task 4), and detection of Type-II hidden nodes along with orientation of the edges to obtain the pattern of the minimal latent polytree (Task 5). Observe that the full polytree is recovered at the end of Task 5 since no edge is left undirected.

Note that since there are no contradictions in the orientation of any edge, no Type-II hidden node is discovered.

4.2.2 A Polytree Network with Only Type-I Hidden Nodes

Consider the 6-node polytree network in Figure 4.4 (True). This network has three rooted subtrees where one root is hidden.

As expected, after applying Task 1, we obtain the lists of nodes belonging to each rooted subtree as in Figure 4.4 (Task 1). Task 2 recovers the collapsed quasi-skeleton of each rooted subtree as in Figure 4.4 (Task 2). The pair of nodes y_3 and y_4 are in common in the neighbors of C_A and C_C, while the pair of nodes y_5 and y_6 are in common in the neighbors of C_B and C_C. Thus, HCMA merges the three hidden clusters together as in Figure 4.4 (Task 3). Observe that at this stage we have $\mathcal{P} = \{C_{A'}\}$ and the neighbors of $C_{A'}$ are considered at the beginning of Task 4 as in Figure 4.4 (Task 4a). When we implement the tests in Theorem 4.5 and Lemma 4.6, we have $I = \{3\}$ (where 3 represents the rooted subtree with root y_{h_1}), $W = \{y_3, y_4, y_5, y_6\}$ and $\overline{W} = \{y_1, y_2, y_3, y_4, y_5, y_6\}$. Then we can create fictitious hidden clusters as explained in Steps 19-23 of HCLA and depicted in Figure 4.4 (Task 4b). Hidden clusters $C_1^{(1)}$ and $C_2^{(1)}$ are then merged together

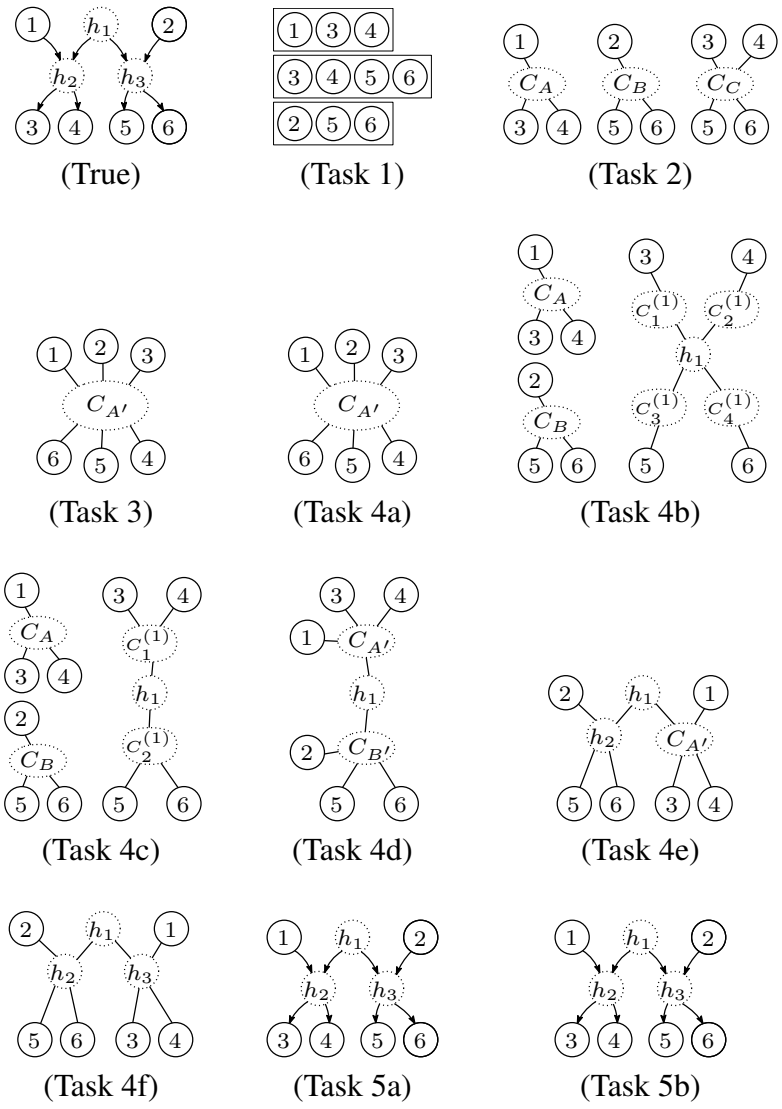


Figure 4.4: The actual minimal latent polytree (True), the lists of visible nodes for each rooted subtree (Task 1), collapsed quasi-skeletons of each rooted subtree (Task 2), merging of the overlapping hidden clusters (Task 3), considering the neighbors of the hidden cluster C'_A (Task 4a), detection of one Type-I hidden node and creation of fictitious hidden clusters (Task 4b), merging of the fictitious hidden clusters (Task 4c), merging of the overlapping hidden clusters (Task 4d), detection of one Type-I hidden node (Task 4e), detection of one Type-I hidden node (Task 4f), detection of Type-II hidden nodes along with orientation of the edges to obtain the pattern of the latent polytree (Task 5a), and no edge has conflicting orientation and therefore no Type-II hidden node is detected (Task 5b). Observe that the full polytree is recovered at the end of Task 5 since no edge is left undirected.

because of the pair y_3 and y_4 and hidden clusters $C_3^{(1)}$ and $C_4^{(1)}$ are then merged together because of the pair y_5 and y_6 as explained in Steps 24-27 of HCLA and depicted in Figure 4.4 (Task 4c).

After this step, we have that $\mathcal{P} = \{C_{A'}, C_{B'}\}$ and HCLA keeps finding the hidden nodes in a similar manner. Note that the result of Task 5 exactly matches the original network structure since no edges are left unoriented.

4.2.3 A Polyforest Network

Consider the 19-node polyforest network in Figure 4.5 (True). This network has seven rooted subtrees that are contained in two different polytrees. Observe that this polyforest contains both Type-I and Type-II hidden nodes. In this example, we show that the PLA is also capable of recovering the pattern of polyforest networks.

A similar step by step process is shown in Figure 4.5 as PLA progresses through the learning process. After applying Task 1, we obtain the lists of nodes belonging to each rooted subtree as in Figure 4.5 (Task 1). Task 2 recovers the collapsed quasi-skeleton of each rooted tree as in Figure 4.5 (Task 2). Observe that the pair y_{10} and y_{11} is common in the neighbors of C_A and C_D, while the pair y_{17} and y_{18} is common in the neighbors of C_B, C_C, C_E and C_F. Thus, HCMA merges the overlapping hidden clusters together as in Figure 4.5 (Task 3). Observe that at this stage we have $\mathcal{P} = \{C_{A'}, C_{B'}\}$. Without loss of generality, HCLA considers $C_{A'}$ and selects the neighbors of $C_{A'}$ at the beginning of Task 4 as in Figure 4.5 (Task 4a).

When we implement the tests in Theorem 4.5 and Lemma 4.6, we have $I = \{2\}$ (where 2 represents the rooted subtree with root y_{h_2}), $W = \{y_7, y_8, y_{10}, y_{11}, y_{12}\}$ and $\overline{W} = \{y_3, y_{10}, y_{11}, y_{12}\}$. Then we can create fictitious hidden clusters as explained in Steps 19-23 of HCLA and depicted in Figure 4.5 (Task 4b). Hidden clusters $C_1^{(1)}$, $C_2^{(1)}$ and $C_3^{(1)}$ are then merged together because of the nodes y_{10}, y_{11} and y_{12} as explained in Steps 24-27 of HCLA and depicted in Figure 4.5 (Task 4c). The two hidden clusters $C_1^{(1)}$ and C_D are clustered together after applying the HCMA as in Figure 4.5 (Task 4d) since they share a pair of visible neighbors, y_{11} and y_{12}.

After this step, we have that $\mathcal{P} = \{C_{A'}, C_{B'}\}$ and HCLA chooses, without loss of generality, the hidden cluster $C_{A'}$ and its negihbors as in Figure 4.5 (Task 4e). HCLA keeps finding the hidden

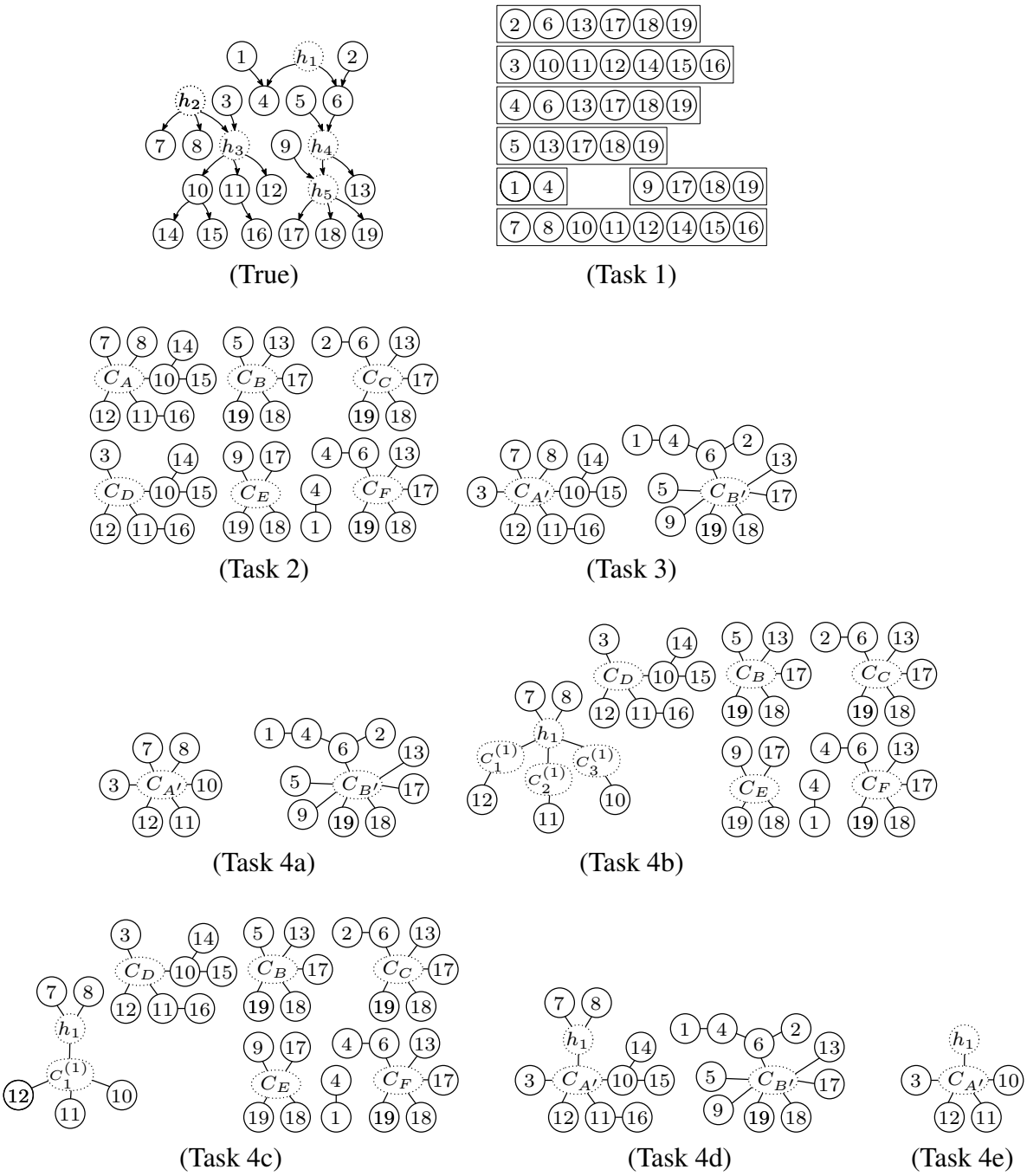


Figure 4.5: The actual minimal latent polyforest (True), the lists of visible nodes for each rooted subtree (Task 1), collapsed quasi-skeletons of each rooted subtree (Task 2), merging of the overlapping hidden clusters (Task 3), considering the neighbors of the hidden cluster $C_{A'}$ (Task 4a), detection of a Type-I hidden node and creating fictitious hidden clusters (Task 4b), merging of the fictitious hidden clusters (Task 4c), merging of the hidden clusters (Task 4d), and considering the neighbors of the hidden cluster $C_{A'}$ (Task 4e).

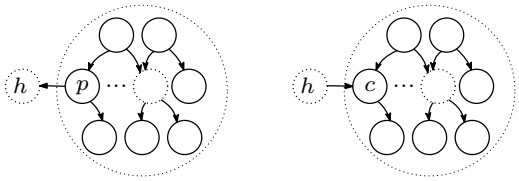
nodes in a similar manner until the polyforest structure is fully learned. Observe that PLA can also recover the structure of polyforest networks due to its nature of working with rooted trees.

4.3 Fundamental Limitations

In this section we show that if a latent polytree P_ℓ is not minimal, then there exists another latent polytree with a smaller number of hidden nodes which has the same independence relations among the visible nodes. In other words, if a latent polytree is not minimal, then there exists at least one hidden node y_h that does not satisfy the minimality conditions (see the degree conditions of Definition 2.21). The proof of such a statement is done by considering various scenarios for such a node [4].

1. Case I: If $\deg_{\vec{P}_\ell}(y_h) = 1$, then this hidden node can be immediately marginalized from the factorization of the joint probability distribution to obtain an equivalent factorization where y_h is not present. Indeed, if $\deg^-_{P_\ell}(y_h) = 1$, then let y_p be the only parent of the node y_h, as depicted in Figure 4.6 (a). Then the factor $\mathbb{P}(y_h \mid y_p)$ disappears from the factorization of the joint probability distributions by integrating over y_h. Instead, if $\deg^+_{P_\ell}(y_h) = 1$, then let y_c be the only child of the node y_h, as depicted in Figure 4.6 (b). Then the factor $\mathbb{P}(y_c \mid y_h)\,\mathbb{P}(y_h)$ disappears from the factorization of the joint probability distributions, again, by integrating over y_h.

2. Case II: If y_h has a single hidden parent y_p and multiple children $y_{c_1}, y_{c_2}, ..., y_{c_{n_c}}$, as depicted in Figure 4.7 (a), then there exists a factor in the factorization of the joint probability distribution



(a) (b)

Figure 4.6: A hidden node y_h where $\deg(y_h) = \deg^-(y_h) = 1$ (a), and a hidden node y_h where $\deg(y_h) = \deg^+(y_h) = 1$ (b) [4].

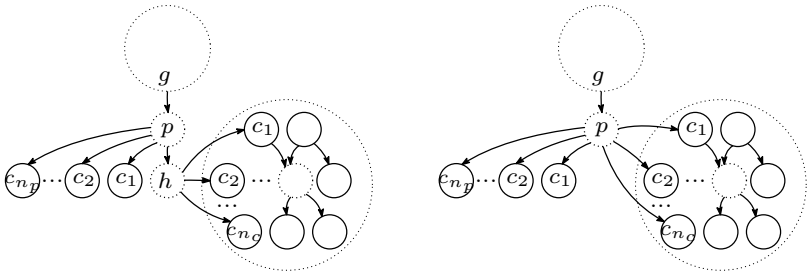


Figure 4.7: A hidden node y_h which has a single hidden parent y_p and multiple children $y_{c_1}, y_{c_2}, ..., y_{c_{n_c}}$ (a), and the case where the hidden node y_h is marginalized (b) [4].

where y_h can be marginalized as follows

$$\prod_{i=1}^{n_c} \mathbb{P}(y_{c_i} \mid y_h, p^{(c_i)})\, \mathbb{P}(y_h \mid y_p) \prod_{j=1}^{n_p} \mathbb{P}(y_{c_j} \mid y_p)\, \mathbb{P}(y_p \mid g) = \prod_{i=1}^{n_c} \mathbb{P}(y_{c_i} \mid y_p, p^{(c_i)}) \prod_{j=1}^{n_p} \mathbb{P}(y_{c_j} \mid y_p)\, \mathbb{P}(y_p \mid g) \tag{4.1}$$

where $p^{(c_i)}$ are the parents of y_{c_i} other than y_h for $i = 1, ..., n_c$, c_j are the children of y_p other than y_h for $j = 1, ..., n_p$ and g are the parents of y_p, as depicted in Figure 4.7 (b).

3. Case III: If y_h is a hidden root with exactly two children y_{c_1} and y_{c_2} and at least one of its children has no other parent (without loss of generality say y_{c_1}), as depicted in Figure 4.8 (a), then in the factorization of the joint probability distribution we find a factor of the following form

$$\prod_{i=1} \mathbb{P}(y_h)\, \mathbb{P}(y_{c_1} \mid y_h)\, \mathbb{P}(y_{c_2} \mid y_h, p) \tag{4.2}$$

where p are the parents of y_{c_2} other than y_h, as depicted in Figure 4.8 (b). By applying Bayes' theorem, we have

$$\mathbb{P}(y_h)\, \mathbb{P}(y_{c_1} \mid y_h)\, \mathbb{P}(y_{c_2} \mid y_h, p) = \mathbb{P}(y_h \mid y_{c_1})\, \mathbb{P}(y_{c_1})\, \mathbb{P}(y_{c_2} \mid y_h, p) \tag{4.3}$$

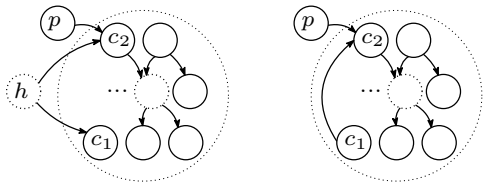


Figure 4.8: A hidden node y_h which is a root with exactly two children y_{c_1} and y_{c_2} and at least one of its children has no other parent (without loss of generality, say y_{c_1}) (a), and the case where the hidden node y_h is marginalized (b) [4].

and then by marginalizing over y_h we obtain the following factor of the joint probability distribution

$$\mathbb{P}(y_{c_1})\,\mathbb{P}(y_{c_2} \mid y_{c_1}, p). \tag{4.4}$$

In all of these scenarios, one hidden node has been marginalized from the factorization of the joint probability distribution of the random variables leading to a factorization equivalent to the original one, but with fewer number of hidden nodes. In all other scenarios, the factorization is instead associated with a polytree which meets the definition of minimality of a latent polytree [4]. Therefore, similar to the case where we assume linear dynamics in the network (see Section 3.4), we have shown that the minimality conditions are necessary and sufficient for learning the structure of a latent polytree network.

Chapter 5

Using Polytrees to Approximate Networks

In this chapter we consider using some of the tools developed in the previous chapters to propose simple and efficient approximators for general networks. As mentioned before, using simpler networks signifies the importance of the chosen links to describe the relationships between the nodes of the network to be approximated which is potentially a more complex network. The approach proposed in this chapter has a polynomial time complexity which makes it favorable for many different applications due to its low computational cost. Although some weak edges are going to be missed during the process of approximation, we show that in the case of high frequency financial data, an example of a real data application, this approximation is meaningful.

5.1 Approximation Algorithm

In this section we describe an algorithm to approximate an LDIM $\mathcal{G}$ using a simpler polytree structure given the observations of the node processes $\{y_i\}_{i=1}^n$ of $\mathcal{G}$. The main idea is to split the process into two steps [1]:

A. Determine the skeleton of the polytree approximating the LDIM structure,

B. Assign orientations to the links of the obtained polytree skeleton.

A schematic representation of the basic steps of this algorithm is given in Figure 5.1. In the following subsections, we explain each step in more detail.

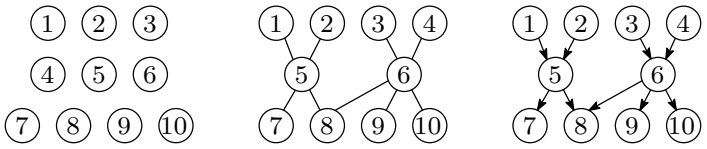


Figure 5.1: Only the observations of the nodes of the actual LDIM are available and the structure is unknown (a), the output of the first step of the approximating algorithm is an undirected tree (b), and in the second step of the algorithm the edges are oriented providing the approximating polytree structure (c) [1].

5.1.1 Step A. Determine the Skeleton of the Approximating Polytree

The main technical tool to determine the skeleton of a polytree approximating the LDIM is the definition of a distance among the processes, as described in Section 3.2.1 and [3]. The log-coherence distance of Equation (2.3) is a function of the power spectral densities of the observed processes $\{y_i\}_{i=1}^n$, thus, under the assumption of ergodicity, it can be estimated directly from their measurements. Furthermore, in the limit of infinite data, the power spectral density estimates are guaranteed to converge to their actual values. Therefore, in the limit of infinite data, the distance of Equation (2.3) can be approximated with arbitrary precision [1].

After estimating the log-coherence distance for every pair of processes y_i and y_j, the skeleton of the approximating polytree is found by computing the Minimum Spanning Tree (MST) over the complete graph (see [76, 80] for definition of MST and complete graph) where the weight of each link is equal to the distance between the corresponding pair of nodes. The Skeleton Approximating Algorithm (SAA), presented in Algorithm 13, provides an algorithmic implementation of this approach for the computation of the skeleton of the approximating polytree. SAA outputs an undirected tree from the knowledge of the log-coherence distances [1].

Algorithm 13 Skeleton Approximating Algorithm

Input a set of nodes $N = \{y_1, ..., y_n\}$ and the distances $d(y_i, y_j)$ for all $y_i, y_j \in N$
Output the undirected tree graph (N, E)

1: Define the complete graph Q over the nodes in N with corresponding weights equal to the distance $d(y_i, y_j)$ for all $y_i, y_j \in N$
2: Apply an MST algorithm to Q and obtain the set of undirected edges E

Observe that the time complexity of SAA depends on the time complexity of the specific MST algorithm. Standard MST algorithms (i.e., Prim or Kruskal) have computational complexity of $n^2 \log(n)$ where n is the number of nodes, but there exist other implementations which are even more efficient [1, 80].

In order to show a fundamental property of SAA, we first introduce the definition of congruity in the skeleton and congruity in the orientations [1].

Definition 5.1 (Congruity in the skeleton and in the orientations)**.** *Consider an algorithm that maps every LDIM $\mathcal{G}$ into a partially directed graph $\bar{G}$. We say that the algorithm is congruous in the skeleton with respect to a set of LDIMs if, for each LDIM in the set, the skeleton of $\bar{G}$ matches the skeleton of the associated graph of $\mathcal{G}$. We say that the algorithm is congruous in the orientations with respect to a set of LDIMs if, for each LDIM in the set, each oriented edge of $\bar{G}$ is in the associated graph of $\mathcal{G}$.*

Now we show an interesting property of *SAA* in the following theorem [1].

Theorem 5.2. *SAA is congruous in the skeleton with respect to the class of LDPTs.*

Proof. The proof is in Appendix D.1. □

5.1.2 Step B. Assign Orientations to the Edges in the Skeleton

Finding a way to assign orientations to the edges of the approximating polytree skeleton, so that the algorithm satisfies congruity in the orientations is a more challenging task. One of the main complicating factors is that when the LDIM to be approximated has a polytree structure, multiple orientations of its edges might still be compatible with the observed data as explained in the following example [1].

Example 1. *Consider an LDIM with two nodes, with the dynamics and power spectral density Φ_u as follows*

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} + \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix}, \qquad \Phi_u = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} \end{pmatrix}. \tag{5.1}$$

The graph associated with this LDIM is depicted in Figure 5.2 (a).

However, the very same three processes y_1, y_2, y_3, could have been generated by the LDIM with the input signals u' and the power spectral density $\Phi_{u'}$ equal to

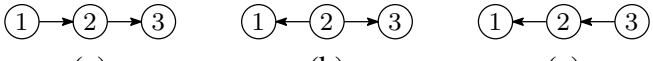
$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} + \begin{pmatrix} u'_1 \\ u'_2 \\ u'_3 \end{pmatrix}, \qquad u' = \begin{pmatrix} \frac{3}{4} & -\frac{1}{2} & 0 \\ \frac{1}{2} & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} u, \qquad \Phi_{u'} = \begin{pmatrix} \frac{1}{2} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{1}{2} \end{pmatrix} \tag{5.2}$$

with its associated graph depicted in Figure 5.2 (b). Since the input signals are not accessible, there is no way to distinguish between the dynamics of Equation (5.1) and Equation (5.2). Furthermore, the processes y_1, y_2, y_3, could have also been generated by the LDIM with the input signals u'' and power spectral density $\Phi_{u''}$ equal to

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} + \begin{pmatrix} u''_1 \\ u''_2 \\ u''_3 \end{pmatrix}, \quad u'' = \begin{pmatrix} \frac{3}{4} & -\frac{1}{2} & 0 \\ \frac{3}{8} & \frac{3}{4} & -\frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & 1 \end{pmatrix} u, \qquad \Phi_{u''} = \begin{pmatrix} \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{5.3}$$

with its associated graph depicted in Figure 5.2 (c).

Example 1 shows that there exist three different LDIMs with one identical skeleton, but different edge orientations which can generate the same output processes $\{y_i\}_{i=1}^n$. Thus, by only observing the outputs, while we can have an algorithm satisfying congruity in the skeleton, in general we cannot have an algorithm capable of directing all edges which at the same time can satisfy congruity in the orientations. Therefore, as it follows from Example 1, the most we can expect from an algorithm assigning orientations to the edges which satisfies congruity in the orientations is that only some of the edges would get an orientation, while the orientation of others



(a) (b) (c)

Figure 5.2: Graph associated with the LDIM in Equation (5.1) (a), graph associated with the LDIM in Equation (5.2) (b), and graph associated with the LDIM in Equation (5.2) (c) [1].

would remain undecided. For this reason, an algorithm congruous in the edge orientations, can only output, in the general case, a partially directed graph [1].

Another interesting observation that we can draw is that there cannot be any LDIM generating the same output processes of Example 1 with structure as the inverted fork of Figure 5.3. Indeed, there is a special property which involves inverted fork configurations in LDPTs. As a special case, Proposition 2.34 states that, in a LDPT, we have that $d_L\left(y_i, y_j\right) = \infty$ when there exists y_k such that $y_i \to y_k \leftarrow y_j$, while, for all other possible combinations of orientations for the edges (namely, $y_i \leftarrow y_k \to y_j$, $y_i \leftarrow y_k \leftarrow y_j$, or $y_i \to y_k \to y_j$), we have that $d_L\left(y_i, y_j\right) \neq \infty$. This property that helps us identify the orientation of the links is formalized in the following corollary [1].

Corollary 5.3. *Let $\vec{P} = (N, \vec{E})$ be a polytree and let $y_i, y_j, y_k \in N$. If y_k is the only node on the path from y_i to y_j, and $d_L\left(y_i, y_j\right) = \infty$, then the link orientation on this path can be fully identified as $y_i \to y_k \leftarrow y_j$.*

Proof. The proof is a direct consequence of Proposition 2.34. □

Thus, when the network to be approximated is known to be a polytree, after obtaining the skeleton, for any three-node path $y_i - y_k - y_j$, in principle, we could test whether $d_L\left(y_i, y_j\right) = \infty$ to determine if y_k is an inverted fork implying that the directions of the edges are necessarily $y_i \to y_k \leftarrow y_j$. We illustrate this idea via an example [1].

Example 2. *Consider an LDIM with structure as in Figure 5.4 (a). From Proposition 2.34 it is immediate to verify that the only three-node paths $y_i - y_k - y_j$ such that $d_L\left(y_i, y_j\right) = \infty$ are*

$$y_1 - y_3 - y_2, \qquad y_5 - y_6 - y_7, \qquad y_5 - y_6 - y_8, \qquad y_7 - y_6 - y_8, \tag{5.4}$$

while all the other three-node paths satisfy $d_L\left(y_i, y_j\right) < \infty$. Given the skeleton of the network, if the distance between each pair of nodes is exactly known, we can obtain the orientation of the edges

Figure 5.3: LDIM with an inverted fork in node y_2 [1].

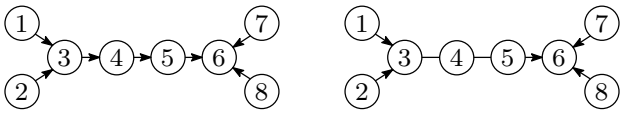


Figure 5.4: Graph associated with an LDIM with 4 inverted forks (a), and the same graph after inferring the link orientations using exact information about the distances (b) [1].

belonging to each of the paths in Equation (5.4). *This strategy provides the partially oriented graph of Figure 5.4 (b), where only the edges* $y_3 - y_4$ *and* $y_4 - y_5$ *are left undirected.*

Hence, Proposition 2.34 can be exploited to detect inverted forks in a polytree. However, Proposition 2.34 can be further used to orient edges that are not involved in inverted forks. Indeed, for any three node path $y_i - y_k - y_j$ in the skeleton, if the direction of the edge $y_i - y_k$ is known to be $y_i \to y_k$ and $d_L\left(y_i, y_j\right) \neq \infty$, Proposition 2.34 implies that the direction of the edge $y_k - y_j$ is $y_k \to y_j$. This is formalized in the following corollary [1].

Corollary 5.4. *Let* $\vec{P} = (N, \vec{E})$ *be a polytree and let* $y_i, y_j, y_k \in N$. *If* y_k *is the only node on the path between* y_i *and* y_j *and also* $(y_i, y_k) \in \vec{E}$, *we have that* $d_L\left(y_i, y_j\right) < \infty$ *if and only if* $(y_k, y_j) \in \vec{E}$.

Proof. The proof is a direct consequence of Proposition 2.34. □

In the following example, we show how we can leverage these two corollaries to infer the orientation of more edges in the approximated polytree [1].

Example 3. *Consider again an LDIM with structure as in Figure 5.4 (a). It was shown in Example 2 how to obtain the partially directed graph of Figure 5.4 (b) where the edges* $y_3 - y_4$ *and* $y_4 - y_5$ *were left undirected. Since the orientation of the edge* $y_1 \to y_3$ *is known and* $d_L(y_1, y_4) < \infty$ *we conclude that* $y_3 - y_4$ *is oriented as* $y_3 \to y_4$. *Now that the orientation of the edge* $y_3 \to y_4$ *is known, we can use this information to infer the orientation of* $y_4 - y_5$. *Indeed, since* $d_L(y_3, y_5) < \infty$ *we necessarily have* $y_4 \to y_5$.

Corollary 5.3 and Corollary 5.4 provide a strategy to orient edges of the skeleton of a polytree approximating an LDIM:

1. find the inverted forks using Corollary 5.3 and obtain a partially oriented polytree $\bar{P} = (N, E, \vec{E})$;

2. propagate the orientations of $\bar{P}$ using Corollary 5.4.

As mentioned in Section 3.2, this approach was proposed by Rebane and Pearl in the context of graphical models in [68]. The Link Orientation Propagation Algorithm (LOPA), presented in Algorithm 14, is an implementation of the technique in [68] that takes as input a partially oriented polytree, with the direction of the v-structures known from Corollary 5.3, and then proceeds to direct the remaining unoriented edges using Corollary 5.4 [1].

Algorithm 14 Link Orientation Propagation Algorithm

Input a partially directed polytree $\bar{P} = (N, E, \vec{E})$ and the distances $d(y_i, y_j)$ for all $y_i, y_j \in N$
Output the partially directed polytree $\bar{P} = (N, E, \vec{E})$

while $\exists\, y_k$ such that $(y_i, y_j) \in \vec{E}$, $(y_j, y_k) \notin \vec{E}$, $\{y_j, y_k\} \in E$, and $d(y_i, y_k) < \infty$ **do**
    set $\vec{E} := \vec{E} \cup \{(y_j, y_k)\}$
end while
for all $\{y_i, y_j\} \in E$ such that $(y_i, y_j) \in \vec{E}$ or $(y_j, y_i) \in \vec{E}$ **do**
    set $E := E \setminus \{\{y_i, y_j\}\}$
end for

One limitation of this approach is that, when dealing with measured data, the test $d_L\left(y_i, y_j\right) = \infty$ cannot be numerically implemented. We would need to replace it with a numerical implementation testing for $d_L\left(y_i, y_j\right) \simeq \infty$. This test might be realized, for example, by choosing an appropriate (i.e., sufficiently large) threshold θ and testing if $d_L\left(y_i, y_j\right) > \theta$. Furthermore, irrespective of the numerical implementation of the test $d_L\left(y_i, y_j\right) \simeq \infty$, its applications on finite time series (also potentially affected by measurement noise), can give rise to contradictory orientations on some edges because of the presence of Type I and Type II errors (namely, false positives and false negatives). Thus, a fundamental problem with this approach is that a naive application of Corollary 5.3 might feed LOPA an input network which has contradictory orientations. The following example demonstrates this issue [1].

Example 4. *Consider again an LDIM with structure as in Figure 5.4 (a). Assume that, because of numerical issues, the only three-node paths $y_i - y_k - y_j$ testing positive for $d_L\left(y_i, y_j\right) \simeq \infty$ are*

$$y_4 - y_5 - y_6, \qquad y_5 - y_6 - y_7, \qquad y_5 - y_6 - y_8, \qquad y_7 - y_6 - y_8,$$

while all the others have tested negative. The test result of the path $y_4 - y_5 - y_6$ is a false positive and the test result of the path $y_1 - y_3 - y_2$ is a false negative. In this scenario, the detected inverted forks create a contradictory orientation on the edge $y_5 - y_6$ as depicted in Figure 5.5.

A straightforward strategy to avoid feeding LOPA an input $\bar{P} = (N, E, \vec{E})$ containing conflicting edge orientations is to assign a significance score to the three-node paths $y_i - y_k - y_j$ testing positive for $d_L\left(y_i, y_j\right) \simeq \infty$. For example, the significance score for the three-node path $y_i - y_k - y_j$ might be given by the estimated log-coherence distance $d_L\left(y_i, y_j\right)$, since a higher value for such an estimate might indicate a higher likelihood that actually $d_L\left(y_i, y_j\right) = \infty$. Once a significance score is chosen, the orientations $y_i \rightarrow y_k \leftarrow y_j$ can be introduced for each triplet $y_i - y_k - y_j$ starting from the ones with higher significance scores so long as they create no conflicting edge orientations, as proposed by the Initial Link Orientation Assignment Algorithm (ILOAA), presented in Algorithm 15 [1].

Algorithm 15 Initial Link Orientation Assignment Algorithm

Input an undirected polytree $P = (N, E)$ and the significance scores $d(y_i, y_j)$ for all $y_i, y_j \in N$
Output partially directed polytree $\bar{P} = (N, E, \vec{E})$

$A := \{(y_i, y_k, y_j) \mid \{y_i, y_k\}, \{y_j, y_k\} \in E, d(y_i, y_j) \simeq \infty\}$
Sort the triplets (y_i, y_k, y_j) in A in decreasing order according to the significance score
while $A \neq \emptyset$ **do**
    let $a = (y_i, y_k, y_j)$ be the first element of A
    remove a from A
    if (y_k, y_i) and (y_k, y_j) are not in $\vec{E}$ **then**
        add (y_i, y_k) and (y_j, y_k) to $\vec{E}$
        remove $\{y_i, y_k\}$ and $\{y_j, y_k\}$ from E
    end if
end while

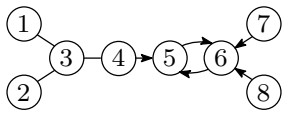


Figure 5.5: The result of inferring edge orientations of the LDIM depicted in Figure 5.4 (a) using information containing Type I and Type II errors [1].

On the other hand, guaranteeing that the input of LOPA has no conflicting orientations is not enough to guarantee that the output of LOPA will not have conflicting orientations. This is shown in the following example [1].

Example 5. *Consider again an LDIM with structure as in Figure 5.4 (a). Assume that, because of numerical issues, the only three-node paths $y_i - y_k - y_j$ testing positive for $d_L\left(y_i, y_j\right) \simeq \infty$ are*

$$y_1 - y_3 - y_2; \qquad y_7 - y_6 - y_8,$$

while all the others have tested negative. Thus, only two inverted forks are detected as shown in Figure 5.6 (a). If we apply LOPA to this scenario, some edges get oriented in both directions, as shown in Figure 5.6 (b).

As demonstrated in Example 5, when approximating an LDIM using a polytree structure, we cannot naively propagate edge orientations as in LOPA, since it might still result in orientation conflicts. This motivates the development of an algorithm analogous to LOPA, but capable of resolving these conflicts. Again, such conflicts can be potentially resolved using several strategies. In the context of this dissertation, we simply propose a modification of LOPA, called Conflict Resolving LOPA (CRLOPA), presented in Algorithm 16, which takes advantage of a significance score to resolve the conflicts in the orientation of the edges. Here, we propose to use the log-coherence distance of Equation (2.3) to calculate the significance score, i.e., the significance score of the triplet $y_i - y_k - y_j$ is equal to $d_L\left(y_i, y_j\right)$ [1].

Note that if the set A is empty, then the output of CRLOPA is trivially the same as the output of LOPA. So, the set A would be empty in cases where there are no contradictions in the edge

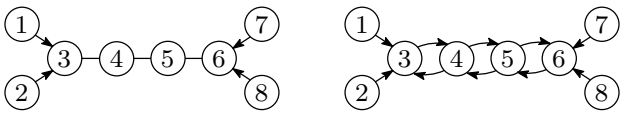


Figure 5.6: The result of inferring edge orientations of the LDIM depicted in Figure 5.4 (a) using information containing errors (a), and the output of LOPA to the graph in Figure 5.6 (a) which results in contradictory link orientations (b) [1].

Algorithm 16 Conflict Resolving LOPA

Input a partially directed polytree $\bar{P} = (N, E, \vec{E})$ and the significance scores $d(y_i, y_j)$ for all $y_i, y_j \in N$

Output the partially directed polytree $\bar{P}$

1: Set $\vec{E}_{perm} := \vec{E}$
2: Set $\bar{P} = (N, E, \vec{E}) := LOPA(\bar{P}, d(\cdot, \cdot))$
3: Define $A := \{(y_i, y_k, y_j) \mid (y_i, y_k) \in \vec{E} \setminus \vec{E}_{temp}, (y_j, y_k) \in \vec{E}\}$
4: **if** $A = \emptyset$ **then**
5: output the partially directed $\bar{P}$
6: **end if**
7: Find the triplet (y_i, y_k, y_j) in A that has the highest significance score
8: Set $\vec{E} := \vec{E}_{perm} \cup \{(y_i, y_k), (y_j, y_k)\}$
9: Go to step 1.

orientations. This would happen, for example, when the actual LDIM has a polytree structure and the distances (i.e., the significance scores) among the nodes are exactly known. In the following proposition we show that if A is empty, then there are no orientation conflicts [1].

Proposition 5.5. *Let $\bar{P} = (N, E, \vec{E})$ be the output of LOPA applied to $(\bar{P}, d(\cdot, \cdot))$. Define A as in Step 3 of CRLOPA. If $A = \emptyset$, then we have that $\forall (y_i, y_j) \in \vec{E} : \nexists (y_j, y_i) \in \vec{E}$.*

Proof. The proof is in Appendix D.2. □

5.1.3 Putting It All Together

Now that we have explained the main idea behind the process of orienting edges in the previous sections, we introduce the Polytree Approximation Algorithm (PAA), presented in Algorithm 17, an algorithm that approximates an LDIM using a polytree [1].

Algorithm 17 Polytree Approximation Algorithm

Input a set of nodes N and the distances $d(y_i, y_j)$ for $y_i, y_j \in N$

Output partially directed polytree $\bar{P} = (N, E, \vec{E})$

1: Set $P = (N, E)$ to the output of SAA applied to the inputs $(N, d(\cdot, \cdot))$
2: Set $\bar{P}_{in} = (N, E, \vec{E})$ to the output of ILOAA applied to $(P, d(\cdot, \cdot))$
3: Set $\bar{P} = (N, E, \vec{E})$ to the output of CRLOPA applied to $(\bar{P}_{in}, d(\cdot, \cdot))$

In the following theorem, we show that PAA is congruous in orienting the links if the original LDIM is a LDPT and the distances are computed exactly [1].

Theorem 5.6. *PAA is congruous in the orientations with respect to the class of LDPTs when the distances between the nodes are computed exactly.*

Proof. The proof is in Appendix D.3. □

5.2 Real Data Application: Stock Market Analysis

In this section, as a benchmark application, we apply our approximation technique to the analysis of a stock portfolio. The dynamics among different stock prices are arguably non-stationary and involve coupling relations with a topology structure which is most likely more complex than a tree. Precisely for these reasons, this application scenario is a challenging benchmark to experiment whether tree structures can be used as adequate approximators for complex networks [1, 23, 40].

We considered the stock prices of the companies listed in the Standard & Poor's 100 (S&P 100) index during normal trading hours in the New York Stock Exchange (NYSE) which happen Monday through Friday, 9:30am till 4:00pm. Data have been sampled every 60 seconds, and for each day we have obtained the time series of the associated logarithmic returns. For each day, we have also computed the log-coherence distance as in Equation (2.3). Following [40], we have averaged these distances over a period of two weeks: 2019/02/25 – 2019/03/08 (Period 1) and determined the skeleton of the tree structure using an MST approach (as explained in SAA). In [40], the average was computed over four weeks, but data were sampled every 120 seconds, so the skeleton approximation method in this article uses the same quantity of data as in [40] with a higher sampling rate [1].

As a fundamental addition to [40], the results presented in this chapter allow one to determine the orientation of the links using ILOAA and CRLOPA. Since we are dealing with finite time series, in order to use these algorithms, we have defined a threshold to determine if the distance between two nodes is close to infinity or not. More specifically, for a three-node path $y_i - y_j - y_k$, we have considered the distance $d_L(y_i, y_k)$ to be infinity when the following equation is satisfied

$$d_L(y_i, y_k) > \max\left(d_L\left(y_i, y_j\right), d_L\left(y_j, y_k\right)\right)(1 + \alpha). \tag{5.5}$$

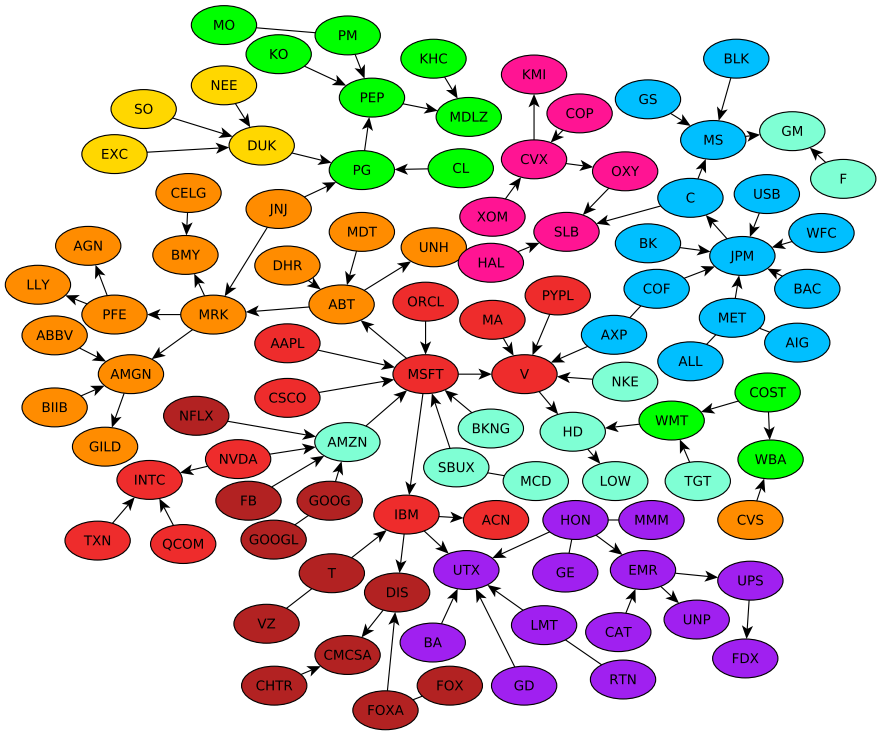
Here, we have arbitrarily chosen $\alpha = 5\%$. The results are shown in Figure 5.7 (a), where the color of the nodes represent the business sector of the companies as given by the Global Industry Classification Standard (GICS). Observe that a tree structure provides a good clustering of the portfolio according to the different business sectors, reproducing the results of [40]. However, we want to go beyond the analysis of [40] and investigate if a tree structure is indeed a good approximator for the unknown network of the portfolio [1].

Since the underlying network is unknown, such a claim is challenging to validate because of the lack of the knowledge of a ground truth to perform the comparison. However, if we assume the existence of a network underlying the portfolio, and if such a network is quasi-stationary or at least slowly varying, we should observe similar patterns/features in the approximating trees when we repeat the same analysis over different time periods. For this reason, we have repeated the same analysis for the additional two-week periods of 2019/03/11 – 2019/03/22 (Period 2), 2019/03/25 – 2019/04/05 (Period 3), and 2019/04/08 – 2019/04/18 (Period 4) (the last period is actually missing one day because NYSE was closed on 2019/04/19, which was Good Friday). The results are shown in Figures 5.7 (b) and 5.8 (a)-(b), respectively [1].

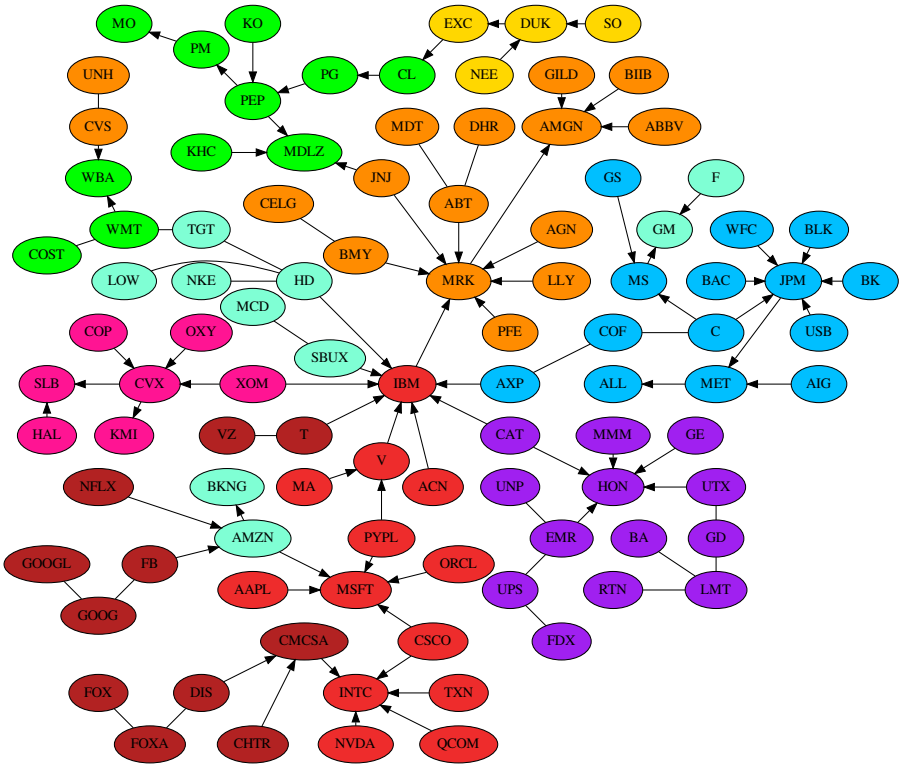
Again, in all of these cases, the identified tree structure provides a good clustering of the different business sectors. To quantify the performance of this technique, we have computed the percentage of inter-cluster links (edges connecting two nodes belonging to the same sector according to GICS) and reported them in Table 5.1 [1]. Observe that such a percentage is overall constant in all the four periods.

A more interesting feature is the percentage of edges which are in common between two consecutive periods and the percentage of edges in common among all the periods, as reported in Table 5.2 [1]. Observe that the two trees associated with two consecutive periods share on average 67% of the edges, and, remarkably, the four trees have 50% of the edges in common. In other words, most of the edges in common between two consecutive periods tend to be shared by all the trees, showing a very solid form of consistency in the recovered structure [1].

It is noteworthy that the edge orientations obtained by application of PAA over the four periods of time also have some similarities in common, especially in terms of nodes with high indegree. By inspecting the orientations obtained after applying this approximation algorithm, we notice that a high indegree indicates a node that is highly correlated with its parents while the parents

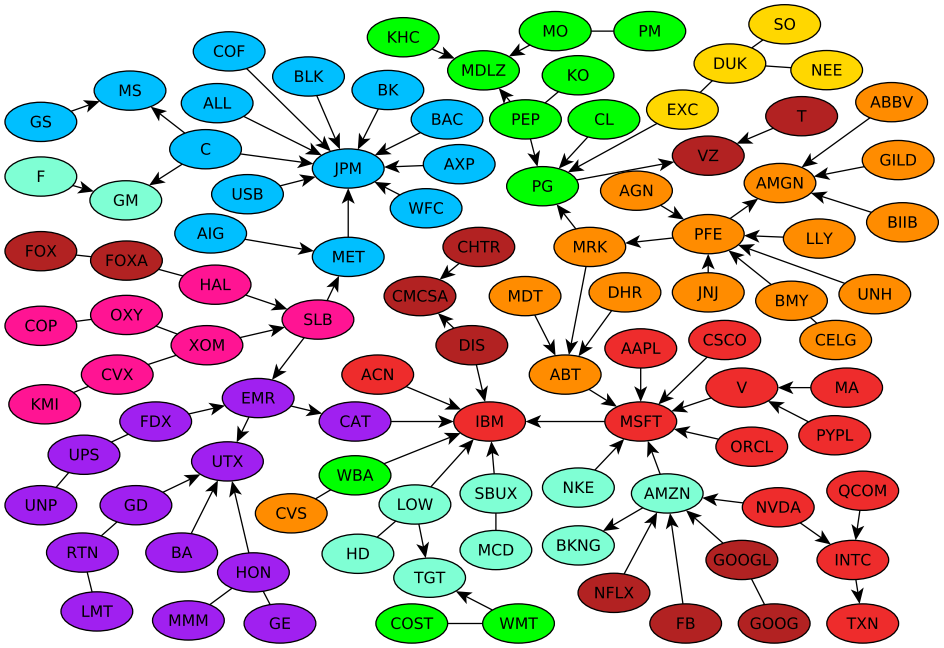


(a)

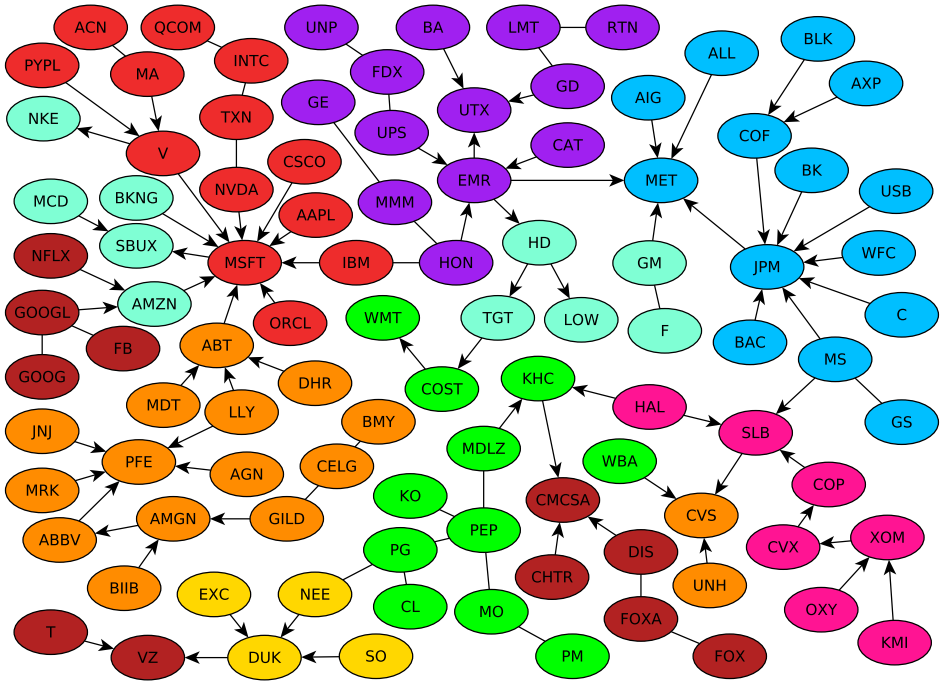


(b)

Figure 5.7: Network structure for Period 1 (a), and network structure for Period 2 (b) [1].



(a)



(b)

Figure 5.8: Network structure for Period 3 (a), and network structure for Period 4 (b) [1].

Table 5.1: Percentage of inter-cluster links in the approximated skeleton [1]

Period 1	Period 2	Period 3	Period 4
78%	83%	78%	80%

Table 5.2: Percentage of common edges in the approximated skeleton [1]

Period 1 & 2	Period 2 & 3	Period 3 & 4	All 4 Periods
73%	63%	62%	50%

are not as correlated with each other. This property would then indicate companies that are contributing in their business areas in a variety of ways, meaning that their stock price behavior tends to correlate with a large number of competitors which focus their activities and attention to a more specific market. While this feature is not perfectly replicated in every period, observe that the nodes MSFT (Microsoft), AMZN (Amazon), IBM (IBM) and JPM (JPMorgan Chase) tend to have consistently high indegrees. Thus, not only the skeleton is capable of displaying meaningful information from the price data in terms of correlated activities between contiguous stocks, but also the edge orientation algorithm can help identify central businesses in the portfolio network by looking for nodes with high degree and, in particular, high indegree [1].

Chapter 6

Conclusions

In this dissertation, we proposed two novel algorithms to learn the structure of a network with a polyforest topology and to infer partial information about the link orientations in the network. These methods have been developed considering the case where some of the nodes might not be observable. It has been shown that if the hidden nodes satisfy some specific degree conditions, then the correct structure, including the location and the number of hidden nodes, is learned. We have also proven that such degree conditions are necessary for the learning process, achieving the fundamental limitations in learning polyforest networks. Moreover, the algorithms developed in this dissertation have polynomial time complexity similar to the methods developed for learning rooted tree structures in the literature. However, the proposed methods here deal with a larger class of networks, namely, polyforests, which can model potential fusion of sources of information in a network. One of the objectives of this dissertation is to develop algorithms that can be applied to both domains of graphical models of random variables and dynamic networks of stochastic processes and it is shown that these two algorithms are applicable to both domains.

We also proposed a novel algorithm to approximate the interconnections of generic networks with simpler polytree networks when the assumption of linear dynamics is exploited. The ultimate goal of this approximation technique is to capture the strongest connections in the network using an algorithm that runs in polynomial time. This scheme is shown to be congruous in the skeleton and orientation of the edges when the network to be approximated has also a polytree structure. Furthermore, we have shown applications of this approximation method to financial market data. The results confirmed that this approach clusters the financial organizations according

to their business sectors showing good agreement with the GICS (Global Industry Classification Standard) classification and it is also confirmed that the inferred orientations represent a sensible interpretation of cause and effect relationships between these organizations.

Moreover, the results of this work could be applied to studies in social sciences. For example, we can model the questions in a survey as a network and then leverage the approaches developed in this dissertation to find the hidden nodes in this network. Thus, we can obtain an analysis of the effectiveness of each question and we can then use this information to design more compelling questions for future studies. Furthermore, another future step would be extending the results developed for tree structured networks to cyclic networks. One viable method would be leveraging junction trees to develop extensions of these learning algorithms to networks containing loops.

Bibliography

[1] F. Sepehr and D. Materassi, "Non-invasive approximation of linear dynamic systems networks using polytrees," *Submitted for review at IEEE Transactions on Control of Network Systems*, 2019. viii, xi, xii, 2, 8, 9, 11, 12, 13, 15, 16, 22, 26, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80

[2] F. Sepehr and D. Materassi, "Inferring the structure of polytree networks of dynamic systems with hidden nodes," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 4618–4623, Dec 2016. ix, 2, 7, 8, 10, 13, 14, 15, 16, 17, 22, 23, 27, 28, 29

[3] F. Sepehr and D. Materassi, "Learning networks of linear dynamic systems with tree topologies," *IEEE Transactions on Automatic Control*, 2019. doi: 10.1109/TAC.2019.2915153. ix, x, xii, 2, 6, 7, 8, 10, 12, 13, 15, 16, 17, 22, 23, 24, 25, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 39, 40, 41, 42, 45, 47, 67, 94, 99, 110

[4] F. Sepehr and D. Materassi, "An algorithm to learn polytree networks with hidden nodes," in *Submitted for review at Neural Information Processing Systems (NIPS)*, 2019. ix, x, xi, xii, 2, 8, 11, 15, 17, 18, 19, 20, 21, 26, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 63, 64, 65, 112, 121, 122

[5] K.-P. Kim, D. Chang, S. K. Lim, S.-K. Lee, H.-K. Lyu, and D.-K. Hwang, "Thermal annealing effects on the dynamic photoresponse properties of al-doped zno nanowires network," *Current Applied Physics*, vol. 11, no. 6, pp. 1311 – 1314, 2011. 1

[6] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D. U. Hwang, "Complex networks: Structure and dynamics," *Physics Reports*, vol. 424, pp. 175–308, February 2006. 1

[7] N. M. Luscombe, M. Madan Babu, H. Yu, M. Snyder, S. A. Teichmann, and M. Gerstein, "Genomic analysis of regulatory network dynamics reveals large topological changes," *Nature*, vol. 431, pp. 308–312, Sep 2004. 1

[8] J. Luo and L. Kuang, "A new method for predicting essential proteins based on dynamic network topology and complex information," *Computational Biology and Chemistry*, vol. 52, pp. 34 – 42, 2014. 1

[9] P. Petousis, S. X. Han, D. Aberle, and A. A. Bui, "Prediction of lung cancer incidence on the low-dose computed tomography arm of the national lung screening trial: A dynamic bayesian network," *Artificial Intelligence in Medicine*, vol. 72, pp. 42 – 55, 2016. 1

[10] S. Anzellotti, D. Kliemann, N. Jacoby, and R. Saxe, "Directed network discovery with dynamic network modelling," *Neuropsychologia*, vol. 99, pp. 1 – 11, 2017. 1

[11] C. N. Kaiser-Bunbury, J. Mougal, A. E. Whittington, T. Valentin, R. Gabriel, J. M. Olesen, and N. Blüthgen, "Ecosystem restoration strengthens pollination network resilience and function," *Nature*, vol. 542, pp. 223–227, Feb 2017. Letter. 1

[12] P. Monestiez, J.-S. Bailly, P. Lagacheriec, and M. Voltz, "Geostatistical modelling of spatial processes on directed trees: Application to fluvisol extent," *Geoderma*, vol. 128, pp. 179–191, 2005. 1

[13] V. M. Carvalho, "From micro to macro via production networks," *The Journal of Economic Perspectives*, vol. 28, no. 4, pp. 23–47, 2014. 1

[14] R. Mantegna and H. Stanley, *An Introduction to Econophysics: Correlations and Complexity in Finance*. Cambridge UK: Cambridge University Press, 2000. 1, 3, 8, 9

[15] M. Nogal, A. O'Connor, B. Caulfield, and B. Martinez-Pastor, "Resilience of traffic networks: From perturbation to recovery via a dynamic restricted equilibrium model," *Reliability Engineering & System Safety*, vol. 156, pp. 84 – 96, 2016. 1

[16] V. Chetty, N. Woodbury, J. Brewer, K. Lee, and S. Warnick, "Applying a passive network reconstruction technique to twitter data in order to identify trend setters," in *2017 IEEE Conference on Control Technology and Applications (CCTA)*, pp. 1344–1349, Aug 2017. 1

[17] A. Dankers, P. M. Van den Hof, X. Bombois, and P. S. Heuberger, "Errors-in-variables identification in dynamic networks - consistency results for an instrumental variable approach," *Automatica*, vol. 62, pp. 39 – 50, 2015. 1, 2

[18] V. Chetty, J. Eliason, and S. Warnick, "Passive reconstruction of non-target-specific discrete-time lti systems," in *2016 American Control Conference (ACC)*, pp. 66–71, July 2016. 1

[19] D. Materassi and M. V. Salapaka, "On the problem of reconstructing an unknown topology via locality properties of the wiener filter," *IEEE Transactions on Automatic Control*, vol. 57, pp. 1765–1777, July 2012. 1, 2, 4, 22, 95

[20] D. Hayden, Y. Yuan, and J. Gonçalves, "Network identifiability from intrinsic noise," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3717–3728, 2017. 1, 2, 3

[21] S. Bolognani, N. Bof, D. Michelotti, R. Muraro, and L. Schenato, "Identification of power distribution network topology via voltage correlation analysis," in *52nd IEEE Conference on Decision and Control*, pp. 1659–1664, Dec 2013. 2, 8

[22] L. Geng and R. Xiao, "Outer synchronization and parameter identification approach to the resilient recovery of supply network with uncertainty," *Physica A: Statistical Mechanics and its Applications*, vol. 482, pp. 407 – 421, 2017. 2

[23] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," *Proceedings of the National Academy of Sciences*, vol. 95, no. 25, pp. 14863–14868, 1998. 2, 3, 8, 9, 76

[24] C. Alexander, *Market models: A guide to financial data analysis*. John Wiley & Sons, 2001. 2

[25] W. R. Shadish, T. D. Cook, D. T. Campbell, *et al.*, *Experimental and quasi-experimental designs for generalized causal inference*. Wadsworth Cengage Learning, 2002. 2

[26] K. Soltesz, K. van Heusden, G. A. Dumont, T. Hägglund, C. L. Petersen, N. West, and J. M. Ansermino, "Closed-loop anesthesia in children using a pid controller: A pilot study," *IFAC Proceedings Volumes*, vol. 45, no. 3, pp. 317–322, 2012. 2nd IFAC Conference on Advances in PID Control. 2

[27] G. Deuschl, C. Schade-Brittinger, P. Krack, J. Volkmann, H. Schäfer, K. Bötzel, C. Daniels, A. Deutschländer, U. Dillmann, W. Eisner, D. Gruber, W. Hamel, J. Herzog, R. Hilker,

S. Klebe, M. Kloß, M. Koy, Jan Krause, A. Kupsch, D. Lorenz, S. Lorenzl, H. M. Mehdorn, J. R. Moringlane, W. Oertel, M. O. Pinsker, H. Reichmann, A. Reuß, G.-H. Schneider, A. Schnitzler, U. Steude, V. Sturm, L. Timmermann, V. Tronnier, T. Trottenberg, L. Wojtecki, E. Wolf, W. Poewe, and J. Voges, "A randomized trial of deep-brain stimulation for parkinson's disease," *New England Journal of Medicine*, vol. 355, no. 9, pp. 896–908, 2006. 2

[28] F. Sepehr and D. Materassi, "On the identification of polytrees of linear dynamic systems with hidden nodes," in *22nd International Symposium on Mathematical Theory of Networks and Systems*, pp. 414–415, July 2016. 2

[29] M. Schmidt, A. Niculescu-Mizil, and K. Murphy, "Learning graphical model structure using l1-regularization paths," in *AAAI*, pp. 1278–1283, AAAI Press, 2007. 2

[30] D. Heckerman, D. Geiger, and D. M. Chickering, "Learning bayesian networks: The combination of knowledge and statistical data," *Machine Learning*, vol. 20, pp. 197–243, Sep 1995. 2

[31] D. Koller and R. Fratkina, "Using learning for approximation in stochastic processes," in *In Proceedings of the International Conference on Machine Learning (ICML)*, pp. 287–295, 1998. 2

[32] M. Sznaier, "Compressive information extraction: A dynamical systems approach," *IFAC Proceedings Volumes*, vol. 45, no. 16, pp. 1559 – 1568, 2012. 16th IFAC Symposium on System Identification. 2

[33] J. Gonçalves and S. Warnick, "Necessary and sufficient conditions for dynamical structure reconstruction of lti networks," *IEEE Transactions on Automatic Control*, vol. 53, no. 7, pp. 1670–1674, 2008. 2, 3, 5

[34] D. Materassi and M. V. Salapaka, "Reconstruction of directed acyclic networks of dynamical systems," in *2013 American Control Conference*, pp. 4687–4692, June 2013. 2, 3, 4

[35] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988. 2, 6, 7, 24

[36] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. The MIT Press, 2009. 2, 12, 20

[37] P. Spirtes, C. Glymour, and R. Scheines, *Causation, Prediction, and Search*. A Bradford Book, 2001. 2, 3, 21

[38] C. Chow and C. Liu, "Approximating discrete probability distributions with dependence trees," *IEEE Transactions on Information Theory*, vol. 14, pp. 462–467, May 1968. 3, 4, 8

[39] E. Mossel, "Distorted metrics on trees and phylogenetic forests," *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, vol. 4, pp. 108–116, Jan. 2007. 3

[40] D. Materassi and G. Innocenti, "Topological identification in networks of dynamical systems," *IEEE Transactions on Automatic Control*, vol. 55, pp. 1860–1871, August 2010. 3, 4, 8, 11, 76, 77, 99

[41] A. Dankers, P. M. J. Van den Hof, X. Bombois, and P. S. C. Heuberger, "Identification of dynamic models in complex networks with prediction error methods: Predictor input selection," *IEEE Transactions on Automatic Control*, vol. 61, pp. 937–952, April 2016. 3

[42] Z. Shen, W.-X. Wang, Y. Fan, Z. Di, and Y.-C. Lai, "Reconstructing propagation networks with natural diversity and identifying hidden sources," *Nature communications*, vol. 5, p. 4323, 2014. 3, 4

[43] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences*, vol. 113, no. 15, pp. 3932–3937, 2016. 3, 4

[44] M. Nitzan, J. Casadiego, and M. Timme, "Revealing physical interaction networks from statistics of collective dynamics," *Science Advances*, vol. 3, no. 2, 2017. 3, 4

[45] M. Dimovska and D. Materassi, "Granger-causality meets causal inference in graphical models: Learning networks via non-invasive observations," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 5268–5273, Dec 2017. 3

[46] M. Eichler, "Granger causality and path diagrams for multivariate time series," *Journal of Econometrics*, vol. 137, no. 2, pp. 334 – 353, 2007. 4

[47] R.-Q. Su, W.-X. Wang, and Y.-C. Lai, "Detecting hidden nodes in complex networks from time series," *Physical review E*, vol. 85, no. 6, p. 065201, 2012. 5

[48] D. Deka, R. Baldick, and S. Vishwanath, "One breaker is enough: hidden topology attacks on power grids," in *Power & Energy Society General Meeting, 2015 IEEE*, pp. 1–5, IEEE, 2015. 5

[49] E. Nozari, Y. Zhao, and J. Cortés, "Network identification with latent nodes via autoregressive models," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 2, pp. 722–736, 2018. 5

[50] E. Kivits and P. M. Van den Hof, "On representations of linear dynamic networks," *IFAC-PapersOnLine*, vol. 51, pp. 838 – 843, 6 2018. 5

[51] Y. Yuan, G.-B. Stan, S. Warnick, and J. Gonçalves, "Robust dynamical network structure reconstruction," *Automatica*, vol. 47, no. 6, pp. 1230–1235, 2011. Special Issue on Systems Biology. 5

[52] Z. Yue, J. Thunberg, W. Pan, L. Ljung, and J. Gonçalves, "Linear dynamic network reconstruction from heterogeneous datasets," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 10586–10591, 2017. 5

[53] M. J. Choi, V. Y. Tan, A. Anandkumar, and A. S. Willsky, "Learning latent tree graphical models," *Journal of Machine Learning Research*, vol. 12, pp. 1771–1812, 2011. 6, 17, 27, 38, 39, 99

[54] M. Zorzi and R. Sepulchre, "Ar identification of latent-variable graphical models," *IEEE Transactions on Automatic Control*, vol. 61, pp. 2327–2340, Sept 2016. 6

[55] A. Anandkumar, K. Chaudhuri, D. J. Hsu, S. M. Kakade, L. Song, and T. Zhang, "Spectral methods for learning multivariate latent tree structure," in *Advances in Neural Information Processing Systems*, pp. 2025–2033, 2011. 6

[56] R. Mourad, C. Sinoquet, N. L. Zhang, T. Liu, and P. Leray, "A survey on latent tree models and applications," *Journal of Artificial Intelligence Research*, vol. 47, pp. 157–203, 2013. 6, 8

[57] Y. Jernite, Y. Halpern, and D. Sontag, "Discovering hidden variables in noisy-or networks using quartet tests," in *Advances in Neural Information Processing Systems 26* (C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, eds.), pp. 2355–2363, Curran Associates, Inc., 2013. 6

[58] P. L. Erdös, M. A. Steel, L. Székely, and T. J. Warnow, "A few logs suffice to build (almost) all trees: Part ii," *Theoretical Computer Science*, vol. 221, no. 1-2, pp. 77–118, 1999. 6

[59] A. Anandkumar and R. Valluvan, "Learning loopy graphical models with latent variables: Efficient methods and guarantees," *The Annals of Statistics*, vol. 41, no. 2, pp. 401–435, 2013. 6

[60] N. Asbeh and B. Lerner, "Learning latent variable models by pairwise cluster comparison: Part i - theory and overview," *Journal of Machine Learning Research*, vol. 17, no. 223, pp. 1–52, 2016. 6

[61] N. Asbeh and B. Lerner, "Learning latent variable models by pairwise cluster comparison: Part ii - algorithm and evaluation," *Journal of Machine Learning Research*, vol. 17, no. 230, pp. 1–45, 2016. 6

[62] J. Pearl, *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2nd ed., 2009. 6, 21

[63] J. Zhang, "On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias," *Artificial Intelligence*, vol. 172, no. 16, pp. 1873 – 1896, 2008. 6, 7

[64] T. Richardson and P. Spirtes, "Ancestral graph markov models," *The Annals of Statistics*, vol. 30, no. 4, pp. 962–1030, 2002. 7

[65] G. Li, A.-M. Vandamme, and J. Ramon, "Learning ancestral polytrees," *Learning Tractable Probabilistic Models*, 2014. The Workshop of Learning Tractable Probabilistic Models at The 31st International Conference on Machine Learning (ICML 2014), Beijing, China. 7

[66] J. Etesami, N. Kiyavash, and T. Coleman, "Learning minimal latent directed information polytrees," *Neural Computation*, vol. 28, no. 9, pp. 1723–1768, 2016. 7, 8

[67] J. Pearl, "Fusion, propagation, and structuring in belief networks," *Artificial Intelligence*, vol. 29, no. 3, pp. 241 – 288, 1986. 8

[68] G. Rebane and J. Pearl, "The recovery of causal polytrees from statistical data," in *Proceedings of the Third Conference on Uncertainty Artificial Intelligence*, pp. 222–228, 1987. 8, 10, 18, 29, 38, 56, 72, 124

[69] M. Ouerd, B. Oommen, and S. Matwin, "A formal approach to using data distributions for building causal polytree structures," *Information Sciences*, vol. 168, no. 1, pp. 111–132, 2004. 8

[70] U. Kjærulff, "Hugs: Combining exact inference and gibbs sampling in junction trees," in *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pp. 368–375, 1995. 8

[71] F. R. Bach and M. I. Jordan, "Thin junction trees," in *Advances in Neural Information Processing Systems*, pp. 569–576, 2002. 8

[72] K. P. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study," in *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, UAI'99, (San Francisco, CA, USA), pp. 467–475, Morgan Kaufmann Publishers Inc., 1999. 8

[73] A. Ujile and Z. Ding, "A dynamic approach to identification of multiple harmonic sources in power distribution systems," *International Journal of Electrical Power & Energy Systems*, vol. 81, pp. 175–183, 2016. 8

[74] D. T. Nguyen, N. P. Nguyen, and M. T. Thai, "Sources of misinformation in online social networks: Who to suspect?," in *MILCOM 2012 - 2012 IEEE Military Communications Conference*, pp. 1–6, Oct 2012. 8

[75] W. Zang, P. Zhang, C. Zhou, and L. Guo, "Discovering multiple diffusion source nodes in social networks," *Procedia Computer Science*, vol. 29, pp. 443–452, 2014. 8

[76] R. Diestel, *Graph Theory*. Springer, 5th ed., 2010. 12, 67, 99, 102, 107, 123

[77] T. Verma and J. Pearl, "Causal Networks: Semantics and Expressiveness," in *Proceedings of the 4th Workshop on Uncertainty in Artificial Intelligence*, pp. 352–359, 1988. 20

[78] E. B. Wilson, "Probable inference, the law of succession, and statistical inference," *Journal of the American Statistical Association*, vol. 22, no. 158, pp. 209–212, 1927. 40

[79] D. Materassi, "Reconstructing tree structures of dynamic systems with hidden nodes under nonlinear dynamics," in *2016 24th Mediterranean Conference on Control and Automation (MED)*, pp. 1331–1336, June 2016. 47, 50, 110, 111, 114

[80] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. The MIT Press, 2nd ed., 2009. 67, 68

[81] T. Kailath, A. H. Sayed, and B. Hassibi, *Linear Estimation*. Prentice Hall, 2000. 94

[82] D. Materassi and M. V. Salapaka, "Notions of separation in graphs of dynamical systems," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 2341–2346, 2014. 95

[83] M. A. Bender, M. Farach-Colton, G. Pemmasani, S. Skiena, and P. Sumazin, "Lowest common ancestors in trees and directed acyclic graphs," *Journal of Algorithms*, vol. 57, no. 2, pp. 75–94, 2005. 100

Appendix

A Proofs Related to Chapter 2

A.1 Proof of Lemma 2.27

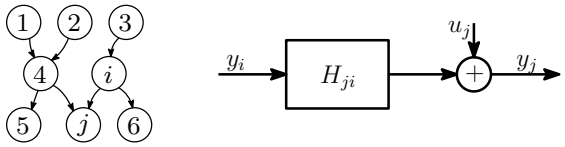
Proof. If $H_{ji}(z) \neq 0$, then there is a path of length 1, namely the edge $(y_i, y_j) \in \vec{E}$. The entry (j, i) of $H^2(z)$ is given by

$$(H^2(z))_{ji} = \sum_{k=1}^{n} H_{jk}(z) H_{ki}(z) \tag{1}$$

and is trivially equal to zero if there is no chain of length 2 from y_i to y_j. Iterating this argument we find that, if there is no chain of length q from y_i to y_j, then $(H^q(z))_{ji} = 0$. Thus, we have $(H^q(z))_{ji} = 0$, for all $q > \ell$ and all y_i, y_j. Now consider the relation $(\mathbf{I} - H(z))\, y = u$ from Equation (2.1). Since $H(z)$ is nilpotent of order $\ell + 1$, the matrix $(\mathbf{I} - H(z))$ is invertible and $(\mathbf{I} - H(z))^{-1} = \mathbf{I} + \sum_{k=1}^{\ell} H^k(z) = T(z)$ which implies that $y = T(z)\, u$. In addition, since there are no directed cycles, it implies that $T_{ii}(z) = 1$. Also, if there is no chain from y_i to y_j where $y_j \neq y_i$, we immediately have $T_{ji}(z) = 0$. Also from Wiener-Khinchin Theorem (see [81]) we have $\Phi_{yu}(z) = T(z)\Phi_u(z)$ which implies $\Phi_{y_j u_i}(z) = T_{ji}(z)\Phi_{u_i}(z) = 0$. □

A.2 Proof of Lemma 2.30

Proof. Consider two directly connected nodes y_i and y_j and their dynamic relation in a block diagram as in Figure A.1. From Wiener-Khinchin Theorem [81], the power spectral and cross-



(a) (b)

Figure A.1: Nodes y_i and y_j in an LDRT (a), and the block diagram of the dynamic relation between the nodes y_i and y_j (b) [3].

spectral densities are

$$\begin{aligned}\Phi_{y_j}(z) &= \Phi_{y_i}(z)|H_{ji}(z)|^2 + \Phi_{u_j}(z) + 2\,\mathrm{Re}(\Phi_{y_i u_j}(z)H_{ji}(z)),\\ \Phi_{y_i y_j}(z) &= \Phi_{y_i}(z)H_{ji}(z)^* + \Phi_{y_i u_j}(z).\end{aligned} \tag{2}$$

From Lemma 2.27 we have that $\Phi_{u_j y_i}(z) = 0$, which leads to

$$C_{ij}(z) = \frac{|\Phi_{y_i}(z)|^2|H_{ji}(z)|^2}{\Phi_{y_i}(z)\left[\Phi_{y_i}(z)|H_{ji}(z)|^2 + \Phi_{u_j}(z)\right]} = \frac{1}{1 + \frac{\Phi_{u_j}(z)}{|H_{ji}(z)|^2\Phi_{y_i}(z)}}. \tag{3}$$

Again, from Lemma 2.27 and for $|z| = 1$, we have that

$$\Phi_{y_i}(z) = \Phi_{u_i}(z) + \sum_{k\neq i} |T_{ik}(z)|^2\Phi_{u_k} \geq \Phi_{u_i}(z) > \eta. \tag{4}$$

Consequently, from the monotonicity of the logarithm, we can bound the logarithm of the coherence, for $|z| = 1$, as follows

$$\log\left|\frac{1}{1 + \frac{\Phi_{u_j}(z)}{|H_{ji}(z)|^2\eta}}\right| < \log\left|C_{ij}(z)\right| < \log\left|\frac{1}{1 + \frac{\eta}{|H_{ji}(z)|^2\Phi_{y_i}(z)}}\right|. \tag{5}$$

Since $\Phi_{u_j}(z)$, $\Phi_{y_i}(z)$ and $H_{ji}(z)$ are rational functions of z with no zeros on the unit circle, the integral of both lower bound and the upper bound are finite and different from zero. Therefore, we have that $0 < d_L\left(y_i, y_j\right) < \infty$. □

A.3 Proof of Proposition 2.33

Proof. Since $\vec{T}$ is a rooted tree, two nodes y_i and y_j are d-separated by y_k (i.e., $dsep \prec y_i, \{y_k\}, y_j \succ_{\vec{T}}$) if and only if y_k is on the unique path connecting y_i and y_j. According to Theorem 8 in [82], $dsep \prec y_i, \{y_k\}, y_j \succ_{\vec{T}}$, implies that the nodes y_i and y_j are Wiener separated by the node y_k. Wiener separation is defined in Definition 21 of [82] where it is stated that two processes y_i and y_j are Wiener separated given y_k if the Wiener filter $(W_{ji}(z), W_{jk}(z))$ used to estimate y_j from y_i and y_k is such that $W_{ji}(z) = 0$. Lemma 26 in [19] states that y_i and y_j are Wiener separated given y_k if and only if the entry (i, j) of the matrix $\Phi^{-1}(z)$ is equal to zero where $\Phi(z)$ is the joint power spectral

density matrix of y_i, y_j and y_k. By inspection, the entry (i, j) of $\Phi^{-1}(z)$ is

$$\left(\Phi^{-1}(z)\right)_{ij} = \frac{\Phi_{ik}(z)\Phi_{kj}(z) - \Phi_{ij}(z)\Phi_{kk}(z)}{A(z)} \tag{6}$$

where $A(z) = \Phi_{ii}(z)\Phi_{jj}(z)\Phi_{kk}(z) + \Phi_{ij}(z)\Phi_{ki}(z)\Phi_{jk}(z) + \Phi_{ji}(z)\Phi_{ik}(z)\Phi_{kj}(z) - \Phi_{ii}(z)\Phi_{jk}(z)\Phi_{kj}(z) - \Phi_{ik}(z)\Phi_{jj}(z)\Phi_{ki}(z) - \Phi_{ij}(z)\Phi_{ji}(z)\Phi_{kk}(z)$. Therefore, setting the numerator of Equation (6) to zero, we get $\forall\omega \in [-\pi, \pi] : C_{ij}(e^{\mathbf{i}\omega}) = C_{ik}(e^{\mathbf{i}\omega})C_{kj}(e^{\mathbf{i}\omega})$, and consequently $\log|C_{ij}(e^{\mathbf{i}\omega})| = \log|C_{ik}(e^{\mathbf{i}\omega})| + \log|C_{kj}(e^{\mathbf{i}\omega})|$, which implies $d_L\left(y_i, y_j\right) = d_L\left(y_i, y_k\right) + d_L\left(y_k, y_j\right)$ (see Equation (2.3)). □

A.4 Proof of Proposition 2.34

We first introduce a lemma and a proposition. The lemma states that the cross-spectral density of nodes that are not related in the associated graph of a topologically identifiable LDPF is zero.

Lemma .1. *Let $\vec{F} = (N, \vec{E})$ be the associated graph of a topologically identifiable LDPF. If $y_i, y_j \in N$ are not related, then $\Phi_{y_i y_j}(z) = 0$.*

Proof. Let $T(z)$ be as defined in Lemma 2.27 and let $y_k \in N$. First assume that $y_k \neq y_i, y_j$. Since y_i and y_j are not related, it is not possible that there is at the same time a chain from y_k to y_i and a chain from y_k to y_j. Then, we have that $T_{ik}(z)T^*_{jk}(z) = 0$. If $k = i$ (or $k = j$), then we have $T_{ii}(z)T^*_{ji}(z) = 0$ (or $T_{ij}(z)T^*_{jj}(z) = 0$) since one is not a descendant of the other. This implies that

$$\Phi_{y_i y_j}(z) = \sum_{p=1}^{n}\sum_{q=1}^{n} T_{ip}(z)\Phi_u(z)T^*_{qj}(z) = 0, \tag{7}$$

where the last equality follows from the fact that $\Phi_u(z)$ is diagonal. □

The following proposition shows how to define an LDRT in an LDPF.

Proposition .2. *Let $\mathcal{F} = (H(z), u)$ be an LDPF with the associated graph $\vec{F} = (N, \vec{E})$ and let $y_r \in N$ be one of its roots. The LDIM $\mathcal{T} = (H^{(T)}(z), u^{(T)})$ which has $\mathrm{de}_{\vec{F}}(y_r)$ as its output processes and the restriction of $\vec{F}$ to $\mathrm{de}_{\vec{F}}(y_r)$ as its associated graph, is in fact an LDRT.*

Proof. Let $N = \{y_1, y_2, ..., y_n\}$ and with no loss of generality, let $y_1 = y_r$ and $\mathrm{de}_{\vec{F}}(y_r) = \{y_1, y_2, ..., y_m\}$. We want to show that $\mathcal{T} = (H^{(T)}(z), u^{(T)})$ is an LDRT with nodes $\{y_1, y_2, ..., y_m\}$.

For any $i, j = 1, ..., m$ define $u_i^{(T)} := u_i + \sum_{k=m+1}^{n} H_{ik} y_k$ and $H_{ji}^{(T)}(z) := H_{ji}(z)$. We first prove that $\Phi_{u_i^{(T)} u_j^{(T)}} = 0$ for distinct $i, j < m$ which implies that $\mathcal{T} = (H^{(T)}(z), u^{(T)})$ is an LDIM.

Consider $p, q > m$ such that $H_{ip}(z) \neq 0$ and $H_{jq}(z) \neq 0$, for distinct $i, j \leq m$. First, we show that y_p and y_q are not related. By contradiction assume they are related. Let $y_{\pi_0}, ..., y_{\pi_\ell}$ be the unique path from y_i to y_j. Observe that there is a chain from y_1 to y_i with all nodes that are descendants of y_1. Analogously, there is a chain from y_1 to y_j with all nodes that are descendants of y_1. Consequently all nodes in the unique path from y_i to y_j are descendants of y_1. Since $H_{ip}(z) \neq 0$ and $H_{jq}(z) \neq 0$, we have that $y_p \neq y_q$ otherwise we would have the path $y_i \leftarrow y_p = y_q \rightarrow y_j$ which is a contradiction. Also, since $H_{ip}(z) \neq 0$ and $H_{jq}(z) \neq 0$, the unique path connecting y_p to y_q has the form $y_p \rightarrow y_{\pi_0} \, ... \, y_{\pi_\ell} \leftarrow y_q$ which implies that the two nodes are not related which is a contradiction. Therefore, according to Lemma .1, we have that $\Phi_{y_p y_q}(z) = 0$. Also, there is no chain from y_j to y_p and no chain from y_i to y_q. Thus, from Lemma 2.27 we have $\Phi_{y_p u_j}(z) = \Phi_{y_q u_i}(z) = 0$. Therefore, for $i \neq j$, we have that $\Phi_{u_i^{(T)} u_j^{(T)}}(z) = 0$ proving that $\mathcal{T} = (H^{(T)}(z), u^{(T)})$ is an LDIM. Define $y^{(T)} = u^{(T)} + H^{(T)}(z)\, y^{(T)}$ and observe by inspection that $y_i^{(T)} = y_i$ for $i \leq m$. Also, observe that the way the nodes have been chosen proves that the associated graph of $\mathcal{T}$ is a rooted tree. □

Now we can incorporate Lemma .1 and Proposition .2 to prove Proposition 2.34.

Proof. $\Leftarrow$: We want to show that if there exists at least one inverted fork on the path from y_i to y_j or there is no path from y_i to y_j, we have $d_L\left(y_i, y_j\right) = \infty$. Let $\Phi_y(z)$ be the power spectral density matrix of the output y. Equation (2.2) and the Wiener-Khinchin Theorem result in

$$\begin{aligned} \Phi_{y_i y_j}(z) &= \left(T(z)\Phi_u(z)T^*(z)\right)_{ij} = \sum_k T_{ik}(z)\left(\Phi_u(z)\right)_{kk} T^*_{kj}(z) \\ &= \sum_k T_{ik}(z)\left(\Phi_u(z)\right)_{kk} \left(T_{jk}(z)\right)^* \end{aligned} \tag{8}$$

where $(\cdot)_{ij}$ denotes the entry (i, j) of a matrix. Because of the inverted fork on the path from y_i to y_j, there cannot be any node y_k such that there is a chain from y_k to y_i and at the same time a chain from y_k to y_j. Therefore, Lemma 2.27 states $(T(z))_{ik} = 0$ and $(T^*(z))_{kj} = 0$ for all k which implies $\Phi_{y_i y_j}(z) = 0$. Therefore, we have $\forall \omega \in [-\pi, \pi] : C_{ij}(\omega) = 0$ and using Equation (2.3), we conclude that $d_L\left(y_i, y_j\right) = \infty$.

$\Rightarrow$: We want to show that if there exists a path from y_i to y_j with no inverted fork, then $d_L\left(y_i, y_j\right) < \infty$. Since y_i and y_j are related, they have a common root ancestor, namely y_r. Using Proposition .2, define the LDRT given by the restriction of $\vec{F}$ to $\mathrm{de}_{\vec{F}}(y_r)$. Note that y_i and y_j are in this restriction and the distance is additive along the paths of this rooted tree according to Proposition 2.33. Therefore, using the result of previous step (i.e., if $(y_p, y_q) \in \vec{E}$, then $d_L\left(y_p, y_q\right) < \infty$) and the fact that there exists exactly one path between any pair of nodes in a rooted tree, we have that $0 < d_L\left(y_i, y_j\right) < \infty$ because $d_L\left(y_i, y_j\right)$ is the sum of positive finite distances between pairs of nodes on the path from y_i to y_j. $\square$

B Proofs Related to Chapter 3

B.1 Proof of Theorem 3.2

Proof. If a hidden node in $\vec{T}$ is RGA-detectable, then it has necessarily degree greater than or equal to 3 in the skeleton of $\vec{T}$. Then the assertion follows from Theorem 5 in [53]. □

B.2 Proof of Proposition 3.3

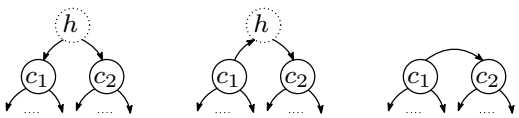
Proof. There exists exactly one root in a rooted tree [76] and by definition, an FD-detectable node is a root. Thus, the only FD-detectable hidden node, namely y_h, is the root of $\vec{T}_\ell$ while all the other hidden nodes are necessarily RGA-detectable. Using Lemma 16 in [40], define an LDRT $\mathcal{T}' = (H(z), u)$ which has the associated graph $\vec{T}'_\ell = (V_T, L_T, \vec{E}')$ with the same skeleton as $\vec{T}_\ell$ and all distances among the nodes in $N_T = V_T \cup L_T$ are the same but the root is y_{c_1} instead of y_h, as in Figures B.1 (a)-(b). For $\mathcal{T}'$ we have

$$\begin{aligned} y_i &= \textstyle\sum_{y_{p_i} \in \mathrm{pa}_{\vec{T}'_\ell}(y_i)} H_{ip_i}(z)\, y_{p_i} + u_i, & y_{c_1} &= u_{c_1}, \\ y_{c_2} &= H_{c_2h}(z) y_h + u_{c_2}, & y_h &= H_{hc_1}(z) y_{c_1} + u_h, \end{aligned} \tag{9}$$

for $y_i \in N_T \setminus \{y_h, y_{c_1}, y_{c_2}\}$, appropriate transfer functions $H_{jk}(z)$ with $y_j, y_k \in N_T$ and mutually independent signals u_j with $y_j \in N_T$.

Now define a new system X such that $x_i := y_i$, $\epsilon_i := u_i$ and $H'_{jk}(z) := H_{jk}(z)$ for all $y_i \in N_T \setminus \{y_h, y_{c_2}\}$, distinct $y_j, y_k \in N_T \setminus \{y_h\}$ where $\{y_j, y_k\} \notin \{y_{c_1}, y_{c_2}\}$, and also

$$x_{c_2} := H'_{c_2c_1}(z) y_{c_1} + \epsilon_{c_2}, \qquad H'_{c_2c_1}(z) := H_{c_2h}(z) H_{hc_1}(z), \qquad \epsilon_{c_2} := u_{c_2} + H_{c_2h}(z) u_h. \tag{10}$$



(a) (b) (c)

Figure B.1: Rooted trees $\vec{T}_\ell$ (a), $\vec{T}'_\ell$ (b), and $\vec{T}_X$ (c) [3].

Since $\Phi_{u_{c_2}u_i}(z) = 0$ and $\Phi_{u_h u_i}(z) = 0$, we have that $\Phi_{\epsilon_{c_2}\epsilon_i}(z) = 0$ for $y_i \in N_T$. Therefore, the associated graph of the system X denoted by (H', ϵ), is

$$T_X = (V_T, L_T \setminus \{y_h\}, \{E_T \cup \{y_{c_1}, y_{c_2}\}\} \setminus \{\{y_h, y_{c_1}\}, \{y_h, y_{c_2}\}\}) \tag{11}$$

as in Figure B.1 (c). Also, all the nodes in $L_T \setminus \{y_h\}$ are RGA-detectable, therefore, according to Theorem 3.2, RGA will output the tree T_X. □

B.3 Proof of Theorem 3.5

We first introduce one definition and two lemmas.

Definition .3 (Lowest common ancestor of a set of nodes [83])**.** *Let $\vec{F} = (N, \vec{E})$ be a polyforest and let Q be a subset of N. Let $\vec{F}_Q$ be the subgraph of $\vec{F}$ restricted to the set of all common ancestors of elements of Q. Define Q_{LCA}, the Lowest Common Ancestors (LCA) of Q, as the set of nodes with outdegree 0 in $\vec{F}_Q$.* □

Lemma .4. *In a rooted tree $\vec{T} = (N, \vec{E})$, for every non-empty set $Q \subseteq N$, there is a unique LCA y_s. It also satisfies*

(i) $Q \subseteq \mathrm{de}_{\vec{T}}(y_s)$,

(ii) $\forall y_t \in \mathrm{de}_{\vec{T}}(y_s) \setminus \{y_s\} : Q \nsubseteq \mathrm{de}_{\vec{T}}(y_t)$.

Proof. For uniqueness of the LCA see section 3 in [83]. First property is an immediate result of Definition .3. Second property holds because otherwise y_t would be the LCA of Q according to Definition .3. □

Lemma .5. *Let $\vec{F} = (N, \vec{E})$ be a polyforest and let $A \subseteq N$. If $\forall y_u, y_v \in A$ we have that y_u and y_v are related, then $\exists y_r : A \subseteq \mathrm{de}_{\vec{F}}(y_r)$.*

Proof. By contradiction assume that for every root y_{r_i}, $\exists y_{v_i} \in A \setminus \mathrm{de}_{\vec{F}}(y_{r_i})$. Let y_r be the root (or one of the roots) for which $\max_i |A \cap \mathrm{de}_{\vec{F}}(y_{r_i})|$ is obtained. Consider the rooted tree with root y_r. Let $Q = A \cap \mathrm{de}_{\vec{F}}(y_r)$ from Lemma .4, then $\exists y_s \in \mathrm{de}_{\vec{F}}(y_r)$ such that y_s is the closest ancestor of the set Q. Therefore, according to the contradiction assumption, $\exists y_v \in A \setminus \mathrm{de}_{\vec{F}}(y_r)$. If there is

no path between y_s and y_v, then there is no path from $y_u \in Q$ to y_v giving the contradiction that y_u and y_v are not related. Thus, there is a path from y_s to y_v in the polyforest $\vec{F}$. Such a path needs to have an inverted fork otherwise y_r would not be in $\arg\max_{y_{r_i}} |A \cap \mathrm{de}_{\vec{F}}(y_{r_i})|$. Let y_{s_1} be the node located immediately after y_s on the path. The path can have the form $y_s \leftarrow y_{s_1} ... \rightarrow y_f \leftarrow ... y_v$ or $y_s \rightarrow y_{s_1} ... \rightarrow y_f \leftarrow ... y_v$ if $y_{s_1} \neq y_f$. If $y_{s_1} = y_f$, then the path has the form $y_s \rightarrow y_f \leftarrow ... y_v$. Consider the following cases:

- if y_s is a child of y_{s_1}, namely $y_u \leftarrow ... \leftarrow y_s \leftarrow y_{s_1} - ... \rightarrow y_f \leftarrow ... \leftarrow y_v$, then choose the node $y_u \in A \cap \mathrm{de}_{\vec{F}}(y_r)$. Therefore, y_u and y_v are not related which is a contradiction;
- if y_{s_1} is a child of y_s, namely $y_u \leftarrow ... \leftarrow y_s \rightarrow y_{s_1} - ... \rightarrow y_f \leftarrow ... \leftarrow y_v$ or $y_u \leftarrow ... \leftarrow y_s \rightarrow y_{s_1} = y_f \leftarrow ... \leftarrow y_v$, then $y_u \in \{A \cap \mathrm{de}_{\vec{F}}(y_r)\} \backslash \mathrm{de}_{\vec{F}}(y_{s_1})$. Therefore, y_u and y_v are not related which is a contradiction.

□

Now we can incorporate previous lemmas and prove Theorem 3.5.

Proof. Let $\vec{F}_\ell = (V, L, \vec{E})$ be the associated graph of the latent LDPF $\mathcal{F} = (H(z), u)$. Consider an arbitrary ordering of the visible nodes to be $y_1, y_2, ..., y_n$. Let S be one of the output lists of PFDA.

If $|S| = 1$, let $S = \{y_j\}$. We know from Step 1 of PFDA that $|S| = 1$ implies that $\forall y \in V \backslash \{y_j\} : d\left(y_j, y\right) = \infty$. Let y_j be such that $y_j \in \mathrm{de}_{\vec{F}_\ell}\left(y_{r_j}\right)$. Therefore, obviously $S \subseteq \mathrm{de}_{\vec{F}_\ell}\left(y_{r_j}\right) \cap V$. Now we show that $S = \mathrm{de}_{\vec{F}_\ell}\left(y_{r_j}\right) \cap V$. By contradiction, assume $\exists y_i \neq y_j : y_i \in \mathrm{de}_{\vec{F}_\ell}\left(y_{r_j}\right) \cap V \backslash S$. Since there are no paths with inverted forks in a rooted tree, we know that y_i and y_j are related, which is a contradiction to $\forall y \in V \setminus \{y_j\} : d\left(y_j, y\right) = \infty$. Thus, we have that $S = \mathrm{de}_{\vec{F}_\ell}\left(y_{r_j}\right) \cap V$.

If $|S| \geq 2$, then let S be the list obtained by PFDA when starting with the pair $\{y_i, y_j\}$ where $d_L\left(y_i, y_j\right) < \infty$. Thus, the output of PFDA is the list $S = S^{(n)}$ where $S^{(k)}$ is defined by the iterations

$$
\begin{aligned}
S^{(0)} &:= \{y_i, y_j\} \\
S^{(k)} &:= \begin{cases} S^{(k-1)} & \text{if } \exists y \in S^{(k-1)} : d_L(y, y_k) = \infty \\ S^{(k-1)} \cup \{y_k\} & \text{otherwise} \end{cases}
\end{aligned} \tag{12}
$$

for $k = 1, ..., n$ as specified in Step 4 of PFDA. We want to show that $\exists y_r : S = \mathrm{de}_{\vec{F}_\ell}(y_r) \cap V$.

- $S \subseteq \text{de}_{\vec{F}_\ell}(y_r) \cap V$: We know that $\forall y_u, y_w \in S$ where $y_u \neq y_w$, we have $d_L(y_u, y_w) < \infty$, and according to Lemma .5 there exists one root y_r such that $S \subseteq \text{de}_{\vec{F}_\ell}(y_r)$. All elements in S are in V, therefore, $S \subseteq \text{de}_{\vec{F}_\ell}(y_r) \cap V$.

- $\text{de}_{\vec{F}_\ell}(y_r) \cap V \subseteq S$: Consider the root y_r in previous step. By contradiction, $\exists y_k \in \text{de}_{\vec{F}_\ell}(y_r) \cap V$ and $y_k \notin S$. Therefore, there exists one vertex $y \in S^{(k-1)}$ such that $d_L(y, y_k) = \infty$. From the previous step $S^{(k-1)} \subseteq S \subseteq \text{de}_{\vec{F}_\ell}(y_r) \cap V$, therefore, $y \in \text{de}_{\vec{F}_\ell}(y_r)$. Since $y_k \in \text{de}_{\vec{F}_\ell}(y_r)$ and there are no paths with inverted forks in a rooted tree, we have that $d(y, y_k) < \infty$, which is a contradiction.

□

B.4 Proof of Theorem 3.6

We introduce five lemmas before providing the proof of Theorem 3.6.

Lemma .6. *In a structurally identifiable latent LDPF with the associated graph $\vec{F}_\ell = (V, L, \vec{E})$, for every non-root node $y \in V \cup L$ if $y \in \text{de}_{\vec{F}_\ell}(y_r)$ for a root node y_r, then $\text{de}_{\vec{F}_\ell}(y_r) \setminus \text{de}_{\vec{F}_\ell}(y) \nsubseteq L$.*

Proof. By contradiction there exists $y \in V \cup L$ such that $y \in \text{de}_{\vec{F}_\ell}(y_r)$ where y_r is a root and $\text{de}_{\vec{F}_\ell}(y_r) \setminus \text{de}_{\vec{F}_\ell}(y) \subseteq L$. Clearly $y_r \neq y$ because y is not a root. The restriction of $\vec{F}_\ell$ to the nodes in $\text{de}_{\vec{F}_\ell}(y_r) \setminus \text{de}_{\vec{F}_\ell}(y) \subseteq L$ is a rooted tree, $\vec{T}$. The tree $\vec{T}$ either is just the root y_r and $y_r \in L$ or has at least two nodes with degree equal to 1 (see [76], Section 1.5). Both cases are in contradiction with $\vec{F}_\ell$ being structurally identifiable because all hidden nodes should have outdegree greater than or equal to 2. □

The following lemma shows that if two root nodes have a common descendant, then the path connecting them contains a unique inverted fork.

Lemma .7. *Let $\vec{F}_\ell = (V, L, \vec{E})$ be the associated graph of a structurally identifiable latent LDPF. If there is a common descendant y for the roots y_{r_i} and y_{r_j} where $y_{r_i}, y_{r_j} \in V \cup L$, then there exists a path from y_{r_i} to y_{r_j} with a unique inverted fork $y_f \in V \cup L$, and $y \in \text{de}_{\vec{F}_\ell}\left(y_f\right)$.*

Proof. Let $y = \text{de}_{\vec{F}_\ell}(y_{r_i}) \cap \text{de}_{\vec{F}_\ell}\left(y_{r_j}\right)$. Therefore, there is a chain from y_{r_i} to y. Consider y_f to be the closest node to y_{r_i} on this chain such that $y_f \in \text{de}_{\vec{F}_\ell}\left(y_{r_j}\right)$. Node y_f exists because there is at least one descendant of y_{r_j} on the chain $y_{r_i} \to \ldots \to y$, namely y. Therefore, we have $y_{r_i} \to \ldots \to y_f \to \ldots \to y$

and $y_{r_j} \to ... \to y_f$. Any node on the chain from y_{r_i} to y_f cannot be on the chain from y_{r_j} to y_f because it would contradict the fact that y_f is a descendant of y_{r_j} that is closest to y_{r_i} on the chain $y_{r_i} \to ... \to y$. Consider the path $y_{r_i} \to ... \to y_f \leftarrow ... \leftarrow y_{r_j}$. This is a valid path in $\vec{F}_\ell$ because all of its nodes are distinct. Therefore, the path from y_{r_i} to y_{r_j} exists and the node y_f is the unique inverted fork on the path from y_{r_i} to y_{r_j}. Also, y is trivially a descendant of y_f in $\vec{F}_\ell$. □

The following lemma guarantees that if a polyforest is structurally identifiable, then the lists obtained by visible descendants of the distinct root nodes are not contained into each other.

Lemma .8. *Let $\vec{F}_\ell = (V, L, \vec{E})$ be the associated graph of a structurally identifiable latent LDPF. The lists obtained by $\mathrm{de}_{\vec{F}_\ell}(y_{r_i}) \cap V$ for all distinct root nodes $y_{r_i} \in V \cup L$ are not contained into each other.*

Proof. By contradiction, assume that $\exists S_i := \{\mathrm{de}_{\vec{F}_\ell}(y_{r_i}) \cap V\} \subseteq S_j := \{\mathrm{de}_{\vec{F}_\ell}\left(y_{r_j}\right) \cap V\}$. Consider the rooted tree T obtained by restricting $\vec{F}$ to $\mathrm{de}_{\vec{F}}(y_{r_i})$. The tree T either has at least two nodes with degree equal to one or it is only the node y_{r_i} in T. If $\mathrm{de}_{\vec{F}_\ell}(y_{r_i}) \cap V = \emptyset$, then we would have that there are at least two hidden nodes with degree equal to one in T or y_{r_i} would be a hidden node itself without any descendants. Both of these situations contradict structural identifiability conditions. Therefore, $\mathrm{de}_{\vec{F}_\ell}(y_{r_i}) \cap V \neq \emptyset$. Since $S_i \subseteq S_j$, the roots y_{r_i} and y_{r_j} have at least one common descendant. According to Lemma .7 there exists a unique inverted fork y_f on the path from y_{r_i} to y_{r_j}. Define $Y_i := \mathrm{de}_{\vec{F}_\ell}(y_{r_i}) \setminus \mathrm{de}_{\vec{F}_\ell}\left(y_f\right)$ and $Y_j := \mathrm{de}_{\vec{F}_\ell}\left(y_{r_j}\right) \setminus \mathrm{de}_{\vec{F}_\ell}\left(y_f\right)$. Obviously, $Y_i \cap \mathrm{de}_{\vec{F}_\ell}\left(y_f\right) = \emptyset$ and $Y_j \cap \mathrm{de}_{\vec{F}_\ell}\left(y_f\right) = \emptyset$. We also have that $Y_i \cap Y_j = \emptyset$, otherwise $y \in Y_i \cap Y_j$ would imply, by Lemma .7, that $y \in \mathrm{de}_{\vec{F}_\ell}\left(y_f\right)$ giving a contradiction with $y \notin \mathrm{de}_{\vec{F}_\ell}\left(y_f\right)$.

Therefore, we can write $S_i = \{\mathrm{de}_{\vec{F}_\ell}\left(y_f\right) \cup Y_i\} \cap V = \{\mathrm{de}_{\vec{F}_\ell}\left(y_f\right) \cap V\} \cup \{Y_i \cap V\}$ and $S_j = \{\mathrm{de}_{\vec{F}_\ell}\left(y_f\right) \cup Y_j\} \cap V = \{\mathrm{de}_{\vec{F}_\ell}\left(y_f\right) \cap V\} \cup \{Y_j \cap V\}$. Using contradiction assumption that $S_i \subseteq S_j$ and the fact that $Y_i \cap Y_j = \emptyset$, we can write $\{Y_i \cap V\} \subseteq \{Y_j \cup V\}$ which implies that $Y_i \cap V = \emptyset$. Therefore, we should have that $Y_i = \mathrm{de}_{\vec{F}_\ell}(y_{r_i}) \setminus \mathrm{de}_{\vec{F}_\ell}\left(y_f\right) \subseteq L$ which is a contradiction because of Lemma .6. □

The following lemma shows that each set of nodes in a rooted tree contains at least one pair of nodes that have their LCA on the path connecting them.

Lemma .9. *Let $\vec{T} = (N, \vec{E})$ be a rooted tree and let $Q \subseteq N$ such that $|Q| \geq 2$. There exists at least one pair of vertices $(y_u, y_w) \in Q$ for which y_s, LCA of Q, is on the unique path from y_u to y_w.*

Proof. By contradiction, assume that for every pair $(y_u, y_w) \in Q$, y_s is not on their unique path. If $y_s \in Q$, we can choose the pair to be (y_u, y_s). Since y_s is on the path from y_u to y_s, this is a contradiction. If $y_s \notin Q$, let $\{y_{c_i}\}_{i=1}^{n_c} = \mathrm{ch}_{\vec{T}}(y_s)$ and consider the following cases.

- If $n_c = 0$, then $\mathrm{de}_{\vec{T}}(y_s) = \{y_s\}$ which implies that $|Q| = 1$ which is a contradiction to $|Q| \geq 2$.
- If $n_c = 1$, then $Q \subseteq \mathrm{de}_{\vec{T}}(y_{c_1})$ which is a contradiction to the property of vertex y_s which states that $\forall y_t \in \mathrm{de}_{\vec{T}}(y_s) \setminus \{y_s\}$, $Q \nsubseteq \mathrm{de}_{\vec{T}}(y_t)$.
- If $n_c \geq 2$, then $\exists y_{c_i}$ such that $Q \cap \mathrm{de}_{\vec{T}}(y_{c_i}) \neq \emptyset$. Indeed, if $Q \cap \mathrm{de}_{\vec{T}}(y_{c_i}) = \emptyset$ for all $i = 1, ..., n_c$, then Q would be empty. Thus, $\exists y_u \in \mathrm{de}_{\vec{T}}(y_{c_i}) \cap Q$. Since $y_{c_i} \in \mathrm{de}_{\vec{T}}(y_s)$, according to the properties of y_s, $Q \nsubseteq \mathrm{de}_{\vec{T}}(y_{c_i})$ and therefore $\exists y_w \in Q \setminus \mathrm{de}_{\vec{T}}(y_{c_i})$. Since $y_w \in \mathrm{de}_{\vec{T}}(y_s)$, there is one chain from y_s to y_w. Since $y_w \notin \mathrm{de}_{\vec{T}}(y_{c_i})$, there exists $y_{c_j} \neq y_{c_i}$ such that $y_w \in \mathrm{de}_{\vec{T}}\left(y_{c_j}\right)$. Thus, we have $y_w \leftarrow ... \leftarrow y_{c_j} \leftarrow y_s \rightarrow y_{c_i} \rightarrow ... \rightarrow y_u$ which implies that the chain from y_s to y_u and the chain from y_s to y_w have only the node y_s in common. Since there is only one path between any two nodes in each rooted tree, y_s is necessarily on the path from y_u to y_w, which is a contradiction to the assumption.

□

The following lemma guarantees that each list containing the visible descendants of a root, which is not contained in any other list, contains a unique pair of visible nodes.

Lemma .10. *Let $\vec{F}_\ell = (V, L, \vec{E})$ be a latent polyforest. Let $S_i = \mathrm{de}_{\vec{F}_\ell}(y_{r_i}) \cap V$ for the root y_{r_i} with $i \in \{1, ..., n_r\}$ where n_r is the number of roots of $\vec{F}_\ell$. Let $|S_i| \geq 2$ and $S_i \nsubseteq S_j$ for every $j \in \{1, ..., n_r\}$ where $i \neq j$. Then there exist $u, w \in S_i$ such that $\{u, w\} \nsubseteq S_j$.*

Proof. By contradiction, assume that $\forall \{y_u, y_w\} \subseteq S_i$ there exists $j \neq i$ such that $\{y_u, y_w\} \subseteq S_j$. Consider the rooted tree obtained by restricting $\vec{F}_\ell$ to $\mathrm{de}_{\vec{F}_\ell}(y_{r_i})$ and choose (y_u, y_w) as in Lemma .9 applied to such a rooted tree with the set $Q = V \cap \mathrm{de}_{\vec{F}_\ell}(y_{r_i})$. Thus, there exists one node y_s on the path from y_u to y_w such that $V \cap \mathrm{de}_{\vec{F}_\ell}(y_{r_i}) \subseteq \mathrm{de}_{\vec{F}_\ell}(y_s)$. Observe that $y_u, y_w \in \mathrm{de}_{\vec{F}_\ell}\left(y_{r_j}\right)$ as well. Consider the restriction of $\vec{F}_\ell$ to $\mathrm{de}_{\vec{F}_\ell}\left(y_{r_j}\right)$ which is a rooted tree, thus there is a unique path from y_u to y_w. Thus, y_s is on that path implying that $y_s \in \mathrm{de}_{\vec{F}_\ell}\left(y_{r_j}\right)$. Then, we have that $y_s \in \mathrm{de}_{\vec{F}_\ell}\left(y_{r_j}\right)$ implying that $\mathrm{de}_{\vec{F}_\ell}(y_s) \subseteq \mathrm{de}_{\vec{F}_\ell}\left(y_{r_j}\right)$. As a consequence, $\mathrm{de}_{\vec{F}_\ell}(y_s) \cap V \subseteq \mathrm{de}_{\vec{F}_\ell}\left(y_{r_j}\right) \cap V$. We know that

$\text{de}_{\vec{F}_\ell}(y_{r_i}) \cap V \subseteq \text{de}_{\vec{F}_\ell}(y_s) \cap V$, and therefore, $\text{de}_{\vec{F}_\ell}(y_{r_i}) \cap V \subseteq \text{de}_{\vec{F}_\ell}\left(y_{r_j}\right) \cap V$ leads to a contradiction because $S_i \nsubseteq S_j$. □

Now we can provide the proof of Theorem 3.6 as follows.

Proof. $\Rightarrow$: Let $S^{(PFDA)}$ be a list generated by PFDA. According to Theorem 3.5, we know that there exists a root node y_r such that $S^{(PFDA)} = \text{de}_{\vec{F}_\ell}(y_r) \cap V$.

$\Leftarrow$: Since $\vec{F}_\ell$ is structurally identifiable, according to Lemma .8 the lists obtained by visible descendants of the distinct root nodes are not contained into each other. Let $S^{(y_r)} = \text{de}_{\vec{F}_\ell}(y_r) \cap V$ for some root y_r.

First we show that if $|S^{(y_r)}| = 1$, then $y_r \in V$ and $\deg^+_{\vec{F}_\ell}(y_r) = 0$. By contradiction, assume that $y_r \notin V$, or $y_r \in V$ and $\deg^+_{\vec{F}_\ell}(y_r) > 0$.

- If $y_r \notin V$, let $y_1, y_2 \in \text{ch}_{\vec{F}_\ell}(y_r)$. Nodes y_1 and y_2 exist because $\vec{F}_\ell$ is structurally identifiable. Since $|\text{de}_{\vec{F}_\ell}(y_r) \cap V| = 1$, we know that $y_1 \in L$ or $y_2 \in L$. Let $y_1 \in L$. This implies that $\deg^+_{\vec{F}_\ell}(y_1) \geq 2$ (because $\vec{F}_\ell$ is structurally identifiable). Thus, in the rooted tree restricted to $\text{de}_{\vec{F}_\ell}(y_1)$ there are at least two nodes with outdegree equal to zero. These nodes have outdegree equal to zero in the polyforest as well. Since $|S^{(y_r)}| = 1$, at least one of these nodes is hidden which leads to a contradiction with structural identifiability conditions.

- If $y_r \in V$ and $\deg^+_{\vec{F}_\ell}(y_r) > 0$, in the rooted tree $\vec{T}$ restricted to $\text{de}_{\vec{F}_\ell}(y_r)$ there is at least one node y_w where $y_w \neq y_r$ with $\deg^+_{\vec{T}}(y_w) = 0$ which implies that $\deg^+_{\vec{F}_\ell}(y_w) = 0$. Since $|\text{de}_{\vec{F}_\ell}(y_r) \cap V| = 1$, we should have $y_w \in L$ which is a contradiction with the polyforest being structurally identifiable.

Therefore, if $|S^{(y_r)}| = 1$ then y_r is not related to any other node implying that $\forall y_i : d(y_r, y_i) = \infty$ where $y_i \neq y_r$. This scenario is considered in Step 1 of *PFDA*.

Now if $|S^{(y_r)}| \geq 2$, according to Lemma .10, $\exists y_u, y_w \in S^{(y_r)}$ and $\{y_u, y_w\}$ is not contained in any other list obtained by $\text{de}_{\vec{F}_\ell}(y_{r_i}) \cap V$ where y_{r_i} are root nodes such that $y_{r_i} \neq y_r$. Let $S^{(PFDA)}$ be the list generated by PFDA starting from (y_u, y_w) under an arbitrary ordering of the vertices in the polyforest. Therefore, $\{y_u, y_w\} \subseteq S^{(PFDA)}$ and according to Theorem 3.5 there exists one root $y_{\hat{r}}$ such that $S^{(PFDA)} = \text{de}_{\vec{F}_\ell}(y_{\hat{r}}) \cap V$. Since $\{y_u, y_w\} \subseteq \text{de}_{\vec{F}_\ell}(y_r) \cap V$ and $\forall y_{r'} \neq y_r$ we have that $\{y_u, y_w\} \nsubseteq \text{de}_{\vec{F}_\ell}(y_{r'}) \cap V$, we necessarily have that $y_{\hat{r}} = y_r$. This implies that $S^{(PFDA)} = S^{(y_r)}$. □

B.5 Proof of Theorem 3.8

We first prove the following lemma.

Lemma .11. *Consider a topologically and structurally identifiable LDPF $\mathcal{F}$ with associated graph $\vec{F}_\ell = (V, L, \vec{E})$. Let y_r be a root of $\vec{F}_\ell$ and $V_T := V \cap \mathrm{de}_{\vec{F}_\ell}(y_r)$. Let $(V_T \cup L_T, E_T)$ and $d_L\left(y_i, y_j\right)$ for $y_i, y_j \in V_T \cup L_T$ be the output of RGA applied to V_T and the distance $d_L\left(y_i, y_j\right)$ for $y_i, y_j \in V_T$. There exists $y_\ell, y_k \in V \setminus V_T$ for $\{y_i, y_j\} \in E_T$ such that $d_L(y_i, y_k) = \infty$, $d_L\left(y_j, y_\ell\right) = \infty$, $d_L(y_i, y_\ell) < \infty$ and $d_L\left(y_j, y_k\right) < \infty$ if and only if there exists an FD-detectable hidden node y_h such that $\mathrm{ch}_{\vec{F}_\ell}(y_h) = \{y_i, y_j\}$.*

Proof. $\Rightarrow$: By contradiction, there is no FD-detectable node y_h between y_i and y_j in the restriction of $\vec{F}_\ell$ to $\mathrm{de}_{\vec{F}_\ell}(y_r)$. Since $\mathcal{F}$ is structurally identifiable and also $\{y_i, y_j\}$ is in the output of RGA applied to V_T and the distance $d_L(y_a, y_b)$ for $y_a, y_b \in V_T$, Proposition 3.3 implies that $\{y_i, y_j\} \in E$ in the restriction of $\vec{F}_\ell$ to $\mathrm{de}_{\vec{F}_\ell}(y_r)$.

Without loss of generality, assume that the link is oriented as $(y_i, y_j) \in \vec{E}$. Since $d_L(y_i, y_\ell) < \infty$, Proposition 2.34 implies that there exists a path with no inverted fork from y_i to y_ℓ. On the other hand, we have $d_L\left(y_j, y_\ell\right) = \infty$ and Proposition 2.34 implies that there exists an inverted fork on the path from y_j to y_ℓ. Thus, y_j cannot be on the path from y_i to y_ℓ. Also, the path from y_ℓ to y_j is the path from y_ℓ to y_i with the addition of link $y_i \to y_j$. Thus, there exists no inverted fork on this path which is a contradiction. Therefore, there exists an FD-detectable node y_h such that $\mathrm{ch}_{\vec{F}_\ell}(y_h) = \{y_i, y_j\}$.

$\Leftarrow$: There exist $y_\ell \neq y_h$ such that $y_\ell \in \mathrm{pa}_{\vec{F}_\ell}(y_i)$, and $y_k \neq y_h$ such that $y_k \in \mathrm{pa}_{\vec{F}_\ell}\left(y_j\right)$. We know that there is no inverted fork on the path from y_i to y_ℓ. Therefore, using Proposition 2.34 we have that $d_L(y_i, y_\ell) < \infty$. Same is true for y_j and y_k, therefore $d_L\left(y_j, y_k\right) < \infty$. Also, we have the path $y_\ell \to y_i \leftarrow y_h \to y_j \leftarrow y_k$. Since there exists an inverted fork on the path from y_ℓ to y_j and an inverted fork on the path from y_i to y_k, Proposition 2.34 implies $d_L\left(y_j, y_\ell\right) = \infty$ and $d_L(y_i, y_k) = \infty$. □

Now we provide the proof of Theorem 3.8.

Proof. We know that all hidden nodes in $\vec{F}_\ell$ are either RGA-detectable or FD-detectable. Theorem 3.2 shows that all the RGA-detectable hidden nodes are identified in Step 1 of HNDA

along with their connected nodes. Lemma .11 shows that all the FD-detectable hidden nodes are identified in Step 3 of HNDA along with their connected nodes. □

B.6 Proof of Proposition 3.9

Proof. If y_h is FD-detectable, then it is a root and therefore appears in exactly one rooted tree. Thus, we only consider the case where y_h is RGA-detectable.

$\Rightarrow$: LDPF $\mathcal{F}$ is structurally identifiable, therefore, $\forall y_h \in L : \deg^+_{\vec{F}_\ell}(y_h) \geq 2$ and $\deg_{\vec{F}_\ell}(y_h) \geq 3$ and let $y_{c_1}, y_{c_2} \in \mathrm{ch}_{\vec{F}_\ell}(y_h)$. This implies that there are at least two nodes with outdegree zero in the restriction $\vec{T}_i$ of $\vec{F}_\ell$ to visible descendants of $y_{h_i} \in L$ (see [76], Section 1.5). Let these nodes be $y_{u_i} \in \mathrm{de}_{\vec{F}_\ell}(y_{c_1}) \cap V$ and $y_{w_i} \in \mathrm{de}_{\vec{F}_\ell}(y_{c_2}) \cap V$. Therefore, we have the chain $y_{h_i} \to y_{c_1} \to \ldots \to y_{u_i}$ and the chain $y_{h_i} \to y_{c_2} \to \ldots \to y_{w_i}$. The nodes on the chain from y_{c_1} to y_{u_i} cannot be on the chain from y_{c_2} to y_{w_i} because if there exists such a node, then there would be two distinct paths from that node to y_{h_i} (one containing y_{c_1} and the other containing y_{c_2}) which is a contradiction to the fact that $\vec{T}_i$ is a rooted tree. Similarly, the nodes on the chain from y_{c_2} to y_{w_i} cannot be on the chain from y_{c_1} to y_{u_i}. Therefore, there exists exactly one path connecting y_{u_i} to y_{w_i} and this path contains y_{h_i}.

Since $y_{h_i} = y_{h_j}$, define $y_u = y_{u_j} = y_{u_i}$ and $y_w = y_{w_j} = y_{w_i}$ where y_{u_j} and y_{w_j} are the counterparts to y_{u_i} and y_{w_i} in the rooted tree $\vec{T}_j$, respectively. Also, we showed that the only path from y_u to y_w contains y_{h_i}. Since $y_{h_i} = y_{h_j}$, we conclude that y_{h_i} exists in the same position on the path from y_u to y_w in tree $\vec{T}_i$ as y_{h_j} exists on the path from y_u to y_w in tree $\vec{T}_j$.

$\Leftarrow$: Since y_{h_i} and y_{h_j} are in the same position on the path from y_u to y_w in $\vec{T}_i$ as on the path from y_u to y_w in $\vec{T}_j$ and there exists at most one path between any pair of nodes in a rooted tree, therefore $y_{h_i} = y_{h_j}$. □

B.7 Proof of Theorem 3.10

Proof. Since $\mathcal{F}$ is structurally identifiable, Theorem 3.6 guarantees that the lists S_i found at Step 1 of PSLA are the visible nodes of each rooted tree in $\vec{F}_\ell$. Theorem 3.8 guarantees that Step 2 of PSLA will identify all hidden nodes in each rooted tree. Also, Proposition 3.9 guarantees that Step 5 of PSLA labels the overlapping hidden nodes in different rooted trees as the same hidden node. Therefore, the combined results of these steps will reconstruct the skeleton of $\vec{F}_\ell$.

Since there are no inverted forks in a rooted tree (see Proposition 2.13) and the distance $d_L\left(y_i, y_j\right)$ is additive along the paths of a rooted tree (see Proposition 2.33), we know that the distance between pairs of nodes in the same rooted tree takes a finite value. These values are computed in Steps 1 and 2 of PSLA. Furthermore, if a pair of nodes is not present in a common rooted tree, then either the nodes are not connected or there exists at least one inverted fork on the path connecting them. Thus, using Proposition 2.34 we know that their distance is infinity. These values are identified in Step 16 of PSLA. Therefore, the output distances of PSLA distinguish between pairs of nodes that have finite or infinite distance between them. □

B.8 Proof of Lemma 3.11

Proof. According to Proposition 2.34, since $d_L\left(y_i, y_j\right) = \infty$, there should be at least one inverted fork on the path from y_i to y_j. This implies that y_k is the inverted fork on this path. □

B.9 Proof of Lemma 3.12

Proof. $\Rightarrow$: According to Proposition 2.34, if $d_L\left(y_i, y_j\right) < \infty$, then there cannot be an inverted fork on the path from y_i to y_j. Therefore, we should have $(y_k, y_j) \in \vec{E}$.

$\Leftarrow$: According to Proposition 2.34, if there is no inverted fork on the path from y_i to y_j, then we have that $d_L\left(y_i, y_j\right) < \infty$. □

B.10 Proof of Theorem 3.13

Proof. The proof is by induction on the number of descendants of y_j.

- Base step: If the number of descendants of y_j equals to 1, then we have that $\mathrm{de}_{\vec{F}}\left(y_j\right) = \{y_j\}$ and $\mathrm{pa}_{\vec{F}}\left(\mathrm{de}_{\vec{F}}\left(y_j\right)\right) = \mathrm{pa}_{\vec{F}}\left(y_j\right)$. In this case, we know that we have $d_L\left(y_i, y_p\right) = \infty$ where $y_p \in \mathrm{pa}_{\vec{F}}\left(y_j\right)$ and $y_p \neq y_i$. Therefore, according to Proposition 2.34, the link $\{y_j, y_p\}$ would be oriented by Step 5 of LOIA.

- Inductive step: Assume that when the number of descendants of y_j is less than or equal to n, the statement holds. When the number of descendants of y_j equals to $n+1$, we know that $d_L\left(y_i, y_p\right) = \infty$ where $y_p \in \mathrm{pa}_{\vec{F}}\left(y_j\right)$ and $y_p \neq y_i$. Therefore, according to Proposition 2.34, the

link $\{y_j, y_p\}$ would be oriented by Step 5 of LOIA. Also, from Proposition 2.34 we know that $d_L(y_i, y_c) < \infty$ (because there is no inverted fork on the path from y_i to y_c) where $y_c \in \text{ch}_{\vec{F}}\left(y_j\right)$. Therefore, according to Proposition 3.12 the link $\{y_j, y_c\}$ would be oriented by Step 8 of LOIA and then LOIA will be recursively applied to a partially directed polyforest where the only oriented link is (y_j, y_c) and y_c is a child of y_j. The number of descendants of y_c is less than or equal to n. Therefore, by applying the induction hypothesis, we have that LOIA will orient all the edges $\{y_k, y_\ell\} \in E$ for $y_k, y_\ell \in \text{de}_{\vec{F}}(y_c) \cup \text{pa}_{\vec{F}}\left(\text{de}_{\vec{F}}(y_c)\right)$. Since Step 5 of LOIA loops over all children of y_j, it will orient all the edges $\{y_k, y_\ell\} \in E$ such that $y_k, y_\ell \in \cup_{y_c \in \text{ch}_{\vec{F}}(y_j)} \text{de}_{\vec{F}}(y_c) \cup \text{pa}_{\vec{F}}\left(\text{de}_{\vec{F}}(y_c)\right)$. Also, we showed that Step 5 will orient all the edges $\{y_k, y_\ell\} \in E$ for $y_k, y_\ell \in y_j \cup \text{pa}_{\vec{F}}\left(y_j\right) \cup \text{ch}_{\vec{F}}\left(y_j\right)$. Thus, we necessarily have that LOIA will orient all the edges $\{y_k, y_\ell\} \in E$ for $y_k, y_\ell \in \text{de}_{\vec{F}}\left(y_j\right) \cup \text{pa}_{\vec{F}}\left(\text{de}_{\vec{F}}\left(y_j\right)\right)$.

□

C Proofs Related to Chapter 4

C.1 Proof of Theorem 4.1

Proof. In Subsection 3.2.1 and [3], it is shown that PFDA outputs the lists of visible nodes belonging to each rooted subtree of the latent polytree $\vec{P}_\ell = (V, L, \vec{E})$ when the distances between pairs of nodes, namely $d(y_i, y_j)$ for $y_i, y_j \in V$, are given by a metric d satisfying the property that $d(y_i, y_j) < \infty$ if and only if y_i and y_j are in the same rooted subtree (see Proposition 2.34). Define $d(y_i, y_j) := 0$ if and only if $\neg\mathcal{I}(y_i, \emptyset, y_j)$, and define $d(y_i, y_j) := \infty$ if and only if $\mathcal{I}(y_i, \emptyset, y_j)$. Using this new metric, the original PFDA in Algorithm 2 becomes the PFDA in Algorithm 7 with all the related guarantees. □

C.2 Explanation of the Reconstruction Algorithm for Latent Rooted Trees

The main goal of the Reconstruction Algorithm for Latent Rooted Trees developed in [79] is to reconstruct the collapsed quasi-skeleton of a latent rooted tree from independence relation of the form $\mathcal{I}(y_i, y_k, y_j)$ or $\neg\mathcal{I}(y_i, y_k, y_j)$ for $y_i, y_j, y_k \in V_r$ where V_r is the set of visible nodes of the rooted tree. The algorithm and its properties are described in detail in [79]. Here we just provide a brief description of the intuition behind it.

In particular, one fundamental result in [79] is that the Reconstruction Algorithm for Latent Rooted Trees can reconstruct the collapsed skeleton of every rooted tree so long as each hidden cluster has degree greater than or equal to 3. All other hidden clusters are undetected: for each hidden cluster with degree equal to 2, the two nodes linked to such a cluster are linked together by the algorithm; for each cluster with degree equal to 1, the algorithm ignores the cluster. In the context of this dissertation, all hidden clusters in each rooted subtree of a minimal latent polytree have degree greater than or equal to 3 with the exception of the special case where we have a hidden root with two visible children. This is basically the main reason why we have introduced quasi-skeletons: in a quasi-skeleton this special case is removed. The following lemma makes sure that hidden clusters in quasi-skeletons of rooted subtrees have degree at least equal to 3 when considering minimal latent polytrees.

Lemma .12. *Let $\vec{P}_\ell = (V, L, \vec{E})$ be a latent polytree and let $T_r = (V_r, L_r, \vec{E}_r)$ be a rooted subtree of $\vec{P}_\ell$ with the root y_r. If $\vec{P}_\ell$ is minimal, then each hidden cluster in the quasi-skeleton of T_r has degree at least* 3.

Proof. Since $\vec{P}_\ell$ is minimal, we distinguish the following two cases:

1. if the hidden node y_h has $\deg^+_{\vec{P}_\ell}(y_h) \geq 2$ and $\deg_{\vec{P}_\ell}(y_h) \geq 3$ then it is trivially true that the hidden cluster that y_h belongs to has degree at least 3 in the quasi-skeleton of the rooted subtree T_r.

2. if the hidden node y_h has $\deg^+_{\vec{P}_\ell}(y_h) = 2$ and $\deg^-_{\vec{P}_\ell}(y_h) = 0$, then we have two subscenarios:

    a. if the two children of y_h are visible, then the hidden cluster containing y_h is made of only y_h. However, y_h is a Type-II hidden node and therefore such a cluster does not appear in the quasi-skeleton of T_r.

    b. if at least one child of y_h is hidden, say y_c, then the hidden cluster containing y_h is the same hidden cluster that contains y_c. Since y_c is a Type-I hidden node, we fall back to case 1, proving that the hidden cluster containing y_h has degree at least 3.

□

Thus, Lemma .12 allows us to apply Reconstruction Algorithm for Latent Rooted Trees to recover the quasi-skeletons of the rooted subtrees.

Now we provide a brief description of the intuition behind the Reconstruction Algorithm for Latent Rooted Trees. Step 1 is just the initialization and Step 2 is a basic induction step solving the problem when the minimal rooted tree has 1 or 2 visible nodes. Observe that, in the collapsed skeleton of a latent rooted tree where all hidden clusters have degree at least 3, all nodes with degree equal to 1 (namely, terminal nodes) are visible and they are either linked to another visible node or linked to a hidden cluster which is connected to at least 2 other visible nodes. Step 3 searches for a terminal visible node y_k: as proven in [79], a node y_k is terminal in a rooted tree where hidden clusters have degree at least 3 if and only if there is no pair of visible nodes $y_i, y_j \in V \setminus \{y_k\}$ such that $\neg\mathcal{I}(y_i, y_k, y_j)$. This is precisely what is tested in this step. For example, considering the polytree in Figure 4.1 (True), one of the lists of visible nodes is the set $V_r = \{y_9, y_{16}, y_{17}, y_{18}\}$ associated with the rooted tree with the root y_9. The quasi-skeleton of this rooted tree is depicted in Figure C.1 (a)

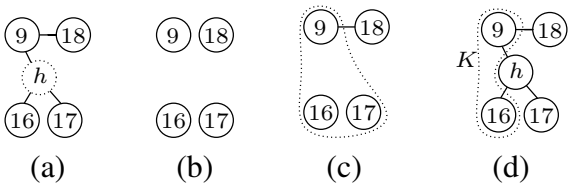


Figure C.1: Quasi-skeleton of the actual rooted tree with root in node y_9 (a), the list of visible nodes belonging to the rooted tree with root in node y_9 (b), node y_{18} satisfies the conditions of being a terminal node and node y_9 satisfies the conditions of being the visible node linked to it (c), and node y_{17} is found to be terminal but not linked to a visible node, thus, a hidden node linked to y_{17} is detected (d) [4].

and the list containing its visible nodes obtained at the end of Task 1 is depicted in Figure C.1 (b). Observe that node y_{18} satisfies the conditions of Step 3 since it cannot d-separate any pair of other nodes in V_r (in other words, the node y_{18} cannot make any pair of visible nodes independent). Thus, node y_{18} is terminal in this rooted subtree.

Once a visible node y_k with $\deg(y_k) = 1$ is found, Step 4 looks for a single visible node y_ℓ linked to y_k. We have that y_ℓ exists if and only if $\forall y_j \in V_r \setminus \{y_k, y_\ell\} : \mathcal{I}(y_k, y_\ell, y_j)$. This is the case for node y_{18}, since we have that y_9 makes y_{18} independent of all the other nodes in V_r. If $y_\ell \in V$ exists, then the test at Step 4 finds it and then at Step 6 the edge $\{y_k, y_\ell\}$ is added to E_r, and the algorithm is run again on $V_r \setminus \{y_k\}$. In our example, the nodes y_9 and y_{18} are linked together and the algorithm is applied to $V_r \setminus \{y_{18}\}$, as depicted in Figure C.1 (c).

When the algorithm runs again on $V_r \setminus \{y_{18}\}$, it is found that, for example, node y_{17} is terminal. However, Step 4 cannot find any visible node linked to y_{17}. Thus, y_{17} must be connected to a hidden cluster: Step 8 is where a new hidden cluster is created. Step 9 finds the set K which contains all other visible neighbors of this hidden cluster. In our example, we have that $K = \{y_9, y_{16}\}$, as depicted in Figure C.1 (d). At Step 10, node y_h is linked to all $y_j \in K$, as depicted in Figure C.1 (d), and the algorithm is applied recursively to $V_r^{(j)} := \{y_i \mid \mathcal{I}(y_k, y_j, y_i)\}$ for all $y_j \in K$ at Step 11. Step 12 sets the output to the union of all the outputs obtained at Step 11.

C.3 Proof of Theorem 4.2

In order to formally prove Theorem 4.2, first we need to introduce two additional results: Theorems .13 and .14. In Theorem .13 a criterion is provided to determine if the unique node linked to a visible terminal node is also visible. This criterion uses only the independence statements involving the visible nodes.

Theorem .13. *Let T be the quasi-skeleton of a rooted subtree of a minimal latent polytree. Let the visible nodes in T be V. Let y_j be a terminal node and let y_k be the unique node linked to y_j. The node y_k is visible if and only if there exists a visible node $y_{k'}$ such that $\mathcal{I}(y_j, y_{k'}, y_i)$ for all $y_i \in V \setminus \{y_j, y_{k'}\}$. Furthermore we have that $y_k = y_{k'}$.*

Proof. $\Rightarrow$: If y_k is visible, then set $y_{k'} = y_k$ and we have that $\mathcal{I}(y_j, y_{k'}, y_i)$ for all $y_i \in V \setminus \{y_j, y_{k'}\}$.

$\Leftarrow$: Let $y_{k'}$ be a visible node such that $\mathcal{I}(y_j, y_{k'}, y_i)$ for all $y_i \in V \setminus \{y_j, y_{k'}\}$. By contradiction, assume that y_k is not visible. Thus, y_k belongs to a hidden cluster and the node y_j is directly linked to such a cluster. Therefore, there are at least two other visible nodes directly linked to this cluster. Let $y_i \neq y_{k'}$ be one of these visible nodes. The path from y_i to y_j involves only hidden nodes, thus it is not true that $\mathcal{I}(y_j, y_{k'}, y_i)$ which is a contradiction. So far, we have shown that the existence of a visible $y_{k'}$ such that $\mathcal{I}(y_j, y_{k'}, y_i)$ for all $y_i \in V \setminus \{y_j, y_{k'}\}$ implies that y_k is visible. We also need to show that $y_{k'} = y_k$. Again, by contradiction, assume that $y_{k'} \neq y_k$. Then, it does not hold that $\mathcal{I}(y_j, y_{k'}, y_k)$ which is a contradiction. □

The following theorem complements Theorem .13 by stating that if there exists a set K of visible nodes that can not be separated from y_j by any other visible nodes and K has at least two elements, then all the nodes in K and y_j are linked to the same hidden cluster.

Theorem .14. *Let T be the quasi-skeleton of a rooted subtree of a minimal latent polytree. Let the visible nodes in T be V. Also, let y_j be a terminal node linked to a hidden cluster C and let K be the set of visible nodes connected to C excluding y_j. Then, it holds that*

$$K = \{y_k \in V \setminus \{y_j\} \mid \forall y_i \in V \setminus \{y_k, y_j\} : \neg\mathcal{I}(y_j, y_i, y_k)\} \tag{13}$$

such that for all $y_i \in V \setminus \{y_k, y_j\}$ there exists $y_k \in K$ such that $\mathcal{I}(y_j, y_k, y_i)$.

Proof. We first show that if $y_k \neq y_j$ is a visible node connected to the hidden cluster C, then $\neg\mathcal{I}(y_j, y_i, y_k)$ for all $y_i \in V \setminus \{y_k, y_j\}$. By contradiction assume that there is $y_i \in V$ such that $\mathcal{I}(y_j, y_i, y_k)$. Now, there is $y_{k'} \in C$ and $y_{j'} \in C$ such that $y_k - y_{k'}$ and $y_j - y_{j'}$ belong to the set of edges E. Since both $y_{j'}$ and $y_{k'}$ belong to cluster C, there is a path from $y_{j'}$ to $y_{k'}$ that does not involve any visible nodes. Thus, there is no $y_i \in V$ such that $\mathcal{I}(y_j, y_i, y_k)$.

Now we show that if $\neg\mathcal{I}(y_j, y_i, y_k)$ for all $y_i \in V \setminus \{y_j, y_k\}$, then y_k is connected to cluster C. By contradiction, assume that it is not. Consider the path from y_j to y_k. Since y_j is a terminal node and it is connected to C, there exists $y_{j'} \in C$ such that $y_j - y_{j'}$ is an edge of E and this is the only edge in the graph involving y_j. Thus, the path from y_j to y_k contains $y_{j'}$. Consider the path from $y_{j'}$ to y_k and let $y_{k'}$ be the first visible node on this path. The node $y_{k'}$ is directly linked to cluster C and if $y_k \neq y_{k'}$, then we have that $\mathcal{I}(y_j, y_{k'}, y_k)$. Thus, we necessarily have that $y_k = y_{k'}$. □

Now, we can provide the proof of Theorem 4.2.

Proof. The proof goes by induction on the number of nodes denoted by n. For $n \leq 2$ the algorithm trivially outputs the correct result. For $n > 2$, we combine Proposition 2.1 and Theorem 4.1 in [79], to guarantee that a visible terminal node y_k is always found at Step 3. Theorem .13 provides a sufficient and necessary condition to find a visible y_ℓ directly linked to y_k. If such a visible node y_ℓ exists, then the edge $y_\ell - y_k$ belongs to the skeleton of the original graph. Then the theorem is applied recursively to a network with $(n-1)$ nodes which is obtained by removing the terminal node y_k from the original one. If such a y_ℓ does not exist, then y_k is necessarily connected to a hidden cluster. Theorem .14 provides a necessary and sufficient condition to find the set K of all visible nodes, other than y_k, linked to such a hidden cluster. A new hidden node y_h is introduced to the set L in order to represent the hidden cluster in the collapsed rooted subtree and the link $y_k - y_h$ is added to the set of edges E. Also the edges $y_h - y_j$ for all $y_j \in K$ are added to E. The algorithm is then applied recursively to each set $V^{(j)}$ given by all visible nodes y_i such that $\mathcal{I}(y_i, y_j, y_k)$. Observe that each $V^{(j)}$ contains fewer number of nodes than n, guaranteeing that the algorithm always terminates. □

C.4 Proof of Theorem 4.3

First we prove a lemma stating that, in a minimal latent polytree, two hidden clusters in two different rooted subtrees share at least one node if and only if they have at least two common neighbors.

Lemma .15. *Let C_1 and C_2 be two distinct hidden clusters in two different rooted subtrees in a minimal latent polytree $\vec{P}_\ell$. The two clusters overlap, i.e., $|C_1 \cap C_2| \geq 1$, if and only if there exist two distinct nodes $y_1, y_2 \in N(C_1) \cap N(C_2)$.*

Proof. $\Rightarrow$: This implication is trivially verified because of the minimality conditions of the latent polytree. If the hidden node in common has two visible descendants y_1 and y_2, then the implication is immediate. If it has, instead, at least one hidden child which belongs to $C_1 \cap C_2$, then such a hidden child either has two visible descendants giving the implication or, again, a hidden child which belongs to $C_1 \cap C_2$. Repeating the argument, we eventually find the common nodes y_1, y_2.

$\Leftarrow$: By contradiction, assume that C_1 and C_2 do not overlap. Since C_1 and C_2 share no common node but have two common neighbors, there must be a loop in the latent polytree $\vec{P}_\ell$, contradicting the fact that it is a polytree. □

Now we can provide the proof of Theorem 4.3.

Proof. From Lemma .15, the proof of Theorem 4.3 is straightforward. If there are two common neighbors, then the two hidden clusters in the two rooted subtrees overlap, thus they belong to the same hidden cluster in the latent polytree. □

C.5 Proof of Theorem 4.4

Proof. The algorithm HCMA proceeds by sequentially merging clusters of the collapsed quasi-skeletons of the rooted subtrees of $\vec{P}_\ell$ if they share at least 2 neighbors (Steps 2-5). According to Lemma .15, this is equivalent to merging these clusters when they overlap (i.e., they have at least one hidden node in common). Thus, the initial set $\mathcal{P}$ contains all the hidden clusters in all the quasi-skeletons of the rooted subtrees of $\vec{P}_\ell$. If two hidden clusters in the quasi-skeletons of two rooted subtrees overlap, then they are necessarily in the same cluster of the quasi-skeleton of the original polytree $\vec{P}_\ell$. Thus, we just need to show that HCMA groups together all the hidden clusters

in quasi-skeletons of the rooted subtrees which are in the same hidden cluster in the quasi-skeleton of $\vec{P}_\ell$.

By contradiction, assume that this is not true. Then the output of HCMA contains a union of clusters that does not exist in the collapsed quasi-skeleton of $\vec{P}_\ell$. Let this union of clusters be U. Thus, there exists at least one hidden node y_h in one hidden cluster C of the quasi-skeleton of $\vec{P}_\ell$ that does not belong to U. Consider the path from y_h to any node in U. By definition such a path consists of all hidden nodes. Let y_a be the last node on such a path that does not belong to U and y_b be the node following y_a on this path. We necessarily have that $y_a \to y_b$, otherwise y_a would be a descendant of y_b and hence in U. Consider a rooted tree containing y_a. Such a rooted tree has a hidden cluster C' which contains y_b as well and consequently overlaps with U, but C' has not been included in U by HCMA. This is a contradiction, because if two clusters overlap, then they are grouped together by HCMA. Therefore, this proves the assertion. □

C.6 Proof of Theorem 4.5

We first provide the following straightforward lemma.

Lemma .16. *Every hidden node in a minimal latent polytree $\vec{P}_\ell = (V, L, \vec{E})$ has at least two visible descendants.*

Proof. Since the latent polytree is minimal, for every hidden node $y_h \in L$ we have that $|\mathrm{ch}_{\vec{P}_\ell}(y_h)| \geq 2$. Now, we distinguish the following two cases.

1. If $|\mathrm{ch}_{\vec{P}_\ell}(y_h) \cap V| \geq 2$, then the statement is trivially true.

2. If $|\mathrm{ch}_{\vec{P}_\ell}(y_h) \cap V| < 2$, then the statement is trivially true by iterating the same argument on one element of the set $\mathrm{ch}_{\vec{P}_\ell}(y_h) \cap L$.

□

Now we leverage the result of Lemma .16 to prove Theorem 4.5.

Proof. $\Rightarrow$: Let $y_{h_r} \in L_r$ be a hidden root of C. Now, we distinguish the following two cases.

- If the root of $\vec{T}_r$, namely y_r, is visible, then y_r is necessarily a parent of y_{h_r}. If the root of $\vec{T}_{r'}$, namely $y_{r'}$, is also a parent of y_{h_r}, then we have $|\tilde{V}_r \setminus \tilde{V}_{r'}| = 1$ and $|\tilde{V}_{r'} \setminus \tilde{V}_r| = 1$ because $\mathrm{de}_C(y_r) \setminus \{y_r\} = \mathrm{de}_C(y_{r'}) \setminus \{y_{r'}\}$. If, instead, the root of $\vec{T}_{r'}$, namely $y_{r'}$, is not a parent of y_{h_r}, then there exists a path connecting one of the children of y_{h_r}, namely y_{c_1}, to $y_{r'}$ and this path necessarily contains an inverted fork. Now, consider another child of y_{h_r}, namely y_{c_2}. Observe that this child exists since all the hidden nodes in the collapsed quasi-skeleton of the rooted subtrees of $\vec{P}_\ell$ have at least outdegree two (see Definitions 2.21 and 2.23). The child node y_{c_2} is either visible itself or has at least two visible descendants according to Lemma .16. If y_{c_2} is visible, then let $A := \{y_{c_2}, y_r\}$. If y_{c_2} is not visible, then let $A := \{\mathrm{de}_C(y_{c_2}) \cap \tilde{V}_r\}$. In either case, we have that $|\tilde{V}_r \setminus \tilde{V}_{r'}| \geq 2$ since $A \cap \mathrm{de}_C(y_{r'}) = \emptyset$ because there exists an inverted fork on the path from y_{h_r} to $y_{r'}$.

- If the root of $\vec{T}_r$, namely y_r, is hidden, then $|\mathrm{ch}_C(y_r)| \geq 2$. If $|\mathrm{ch}_C(y_r)| \geq 3$, let y_{c_1} be the child of y_r such that it is on the path from y_r to $y_{r'}$. Observe that this path contains at least one inverted fork. Thus, if y_{c_2} and y_{c_3} are visible, then we have that $|V_r \setminus V_{r'}| \geq 2$. If, instead any of y_{c_2} or y_{c_3} are hidden, then they should have at least two visible descendants according to Lemma .16 which also results in having $|\tilde{V}_r \setminus \tilde{V}_{r'}| \geq 2$. On the other hand, if $|\mathrm{ch}_C(y_r)| = 2$, then both of these children are hidden since we are working with the collapsed quasi-skeleton of $\vec{P}_\ell$. In this case, each of these hidden children have at least two visible descendants according to Lemma .16. Therefore, we have $|\tilde{V}_r \setminus \tilde{V}_{r'}| \geq 2$.

$\Leftarrow$: We prove, instead, that if $\vec{T}_r$ does not contain a hidden root of C, then $\exists \tilde{V}_{r'} : |\tilde{V}_r \setminus \tilde{V}_{r'}| \leq 1$ and $|\tilde{V}_{r'} \setminus \tilde{V}_r| > 1$. We distinguish the following two cases.

- If the root of $\vec{T}_r$, namely y_r, is visible, then we know that y_r has exactly one hidden child, namely y_{h_c}, because $\vec{T}_r$ belongs to $N(C)$. Since this node is not a hidden root of C, then it has at least one hidden parent, namely y_{h_p}. Let $\vec{T}_{r'}$ be the rooted tree such that $y_{h_p} \in L_{r'}$. In this case, we know that $\tilde{V}_r = \{y_r \cup \mathrm{de}_C(y_{h_c})\}$, $\mathrm{de}_C(y_{h_c}) \subset \tilde{V}_{r'}$, and $y_r \notin \tilde{V}_{r'}$. Thus, we have that $|\tilde{V}_r \setminus \tilde{V}_{r'}| = |\{y_r\}| = 1$.

  Furthermore, if y_{h_p} is the root of $\vec{T}_{r'}$, then this case is similar to the second case of the first part of this proof where there exists a path from y_r to y_{h_p} which contains at least one inverted fork. In this case, we have that $|\tilde{V}_{r'} \setminus \tilde{V}_r| > 1$. If, instead, y_{h_p} is not the root of $\vec{T}_{r'}$, then y_{h_p} has at least two

children because of minimality conditions and at least one parent. The other child of y_{h_p} (i.e., not the node y_{h_c}) and one of the parents of y_{h_p} are either visible or hidden that satisfy minimality conditions. In either case, we have that $|\tilde{V}_{r'} \setminus \tilde{V}_r| > 1$.

- If the root of $\vec{T}_r$ is hidden, then this would be a contradiction to the hypothesis since $\vec{T}_r$ does not contain a hidden root of C.

□

C.7 Proof of Lemma 4.6

First, we introduce a lemma which provides the conditions for finding the parents of a hidden root of a hidden cluster in a minimal latent polytree.

Lemma .17. *Let $\vec{P}_\ell$ be a minimal latent polytree and define the rooted subtrees $\vec{T}_i$, the sets $\tilde{V}_i$ for $i = 1, ..., n_r$, the hidden root y_h and the hidden cluster C as in Lemma 4.6. Let $\tilde{V}_r$ contain y_h which is a hidden root of the hidden cluster C. We have that $\tilde{V}_r \setminus \tilde{V}_{r'} = \{y_v\}$ and $\tilde{V}_{r'} \setminus \tilde{V}_r = \{y_{v'}\}$ if and only if y_v and $y_{v'}$ are parents of y_h.*

Proof. $\Rightarrow$: We show that y_v and $y_{v'}$ are the parents of the root of the hidden cluster C. Let y_r and $y_{r'}$ be the roots of the restriction of $\vec{T}_r$ and $\vec{T}_{r'}$ to the closure of C, respectively. Consider the path from y_r to $y_{r'}$. This path needs to have a length of at least 2, otherwise either y_r or $y_{r'}$ would be a child of the other contradicting the fact that they are roots in the restriction of $\vec{P}_\ell$ to the closure of C. If the length of this path is greater than 2, then it needs to have the form $y_r \rightarrow y_{h_1} \cdots y_{h_2} \leftarrow y_{r'}$ where y_{h_1} and y_{h_2} are two distinct hidden nodes. As a result, either y_{h_1} or y_{h_2} is not a descendant of the other. Furthermore, because of the minimality conditions, in the closure of C, either there exist two visible descendants of y_{h_1} that are not descendants of $y_{r'}$ or there exist two visible descendants of y_{h_2} that are not descendants of y_r. This contradicts the fact that $|\tilde{V}_r \setminus \tilde{V}_{r'}| = |\tilde{V}_{r'} \setminus \tilde{V}_r| = 1$. Thus, the path between y_r and $y_{r'}$ necessarily has length 2 and has the form $y_r \rightarrow y_{h_1} \leftarrow y_{r'}$ for some hidden node y_{h_1}. As a consequence, $y_r = y_v$, $y_{r'} = y_{v'}$ and also $y_{h_1} = y_h$ is the root of the hidden cluster C.

$\Leftarrow$: This implication is trivial. □

Now we can provide the proof of Lemma 4.6.

Proof. For a fixed $\tilde{V}_r$, the set of indices $I \subseteq \{1, 2, ..., n_r\}$ with n_r equal to the number of rooted subtrees is defined as the set $\{r\} \cup \{r'$ such that $|\tilde{V}_r \setminus \tilde{V}_{r'}| = |\tilde{V}_{r'} \setminus \tilde{V}_r| = 1\}$. It is trivial to show that if $|\tilde{V}_r \setminus \tilde{V}_{r'}| = |\tilde{V}_{r'} \setminus \tilde{V}_r|$, then the two sets $\tilde{V}_r$ and $\tilde{V}_{r'}$ can be written as

$$\tilde{V}_r = \{y_v\} \cup \underline{\tilde{V}}, \qquad \tilde{V}_{r'} = \{y_{v'}\} \cup \underline{\tilde{V}}. \tag{14}$$

In other words, there is exactly one element y_v in $\tilde{V}_r$ which is not in $\tilde{V}_{r'}$. Similarly there exists exactly one $y_{v'}$ in $\tilde{V}_{r'}$ which is not in $\tilde{V}_r$.

Now, we show that $W \setminus \overline{W}$ is the set of all nodes linked to y_h which is a root of the hidden cluster C. If a visible node y_w is linked to y_h, it is either its parent or its child. If y_w is a parent of y_h, then it is contained in W and cannot be in $\overline{W}$ because of Lemma .17. If y_w is a child of y_h, then it cannot be in $\overline{W}$ because I contains no index associated with subtrees containing any of the parents of y_h (see Lemma .17).

Now, we show, instead, that if $y_w \in W \setminus \overline{W}$, then y_w is linked to y_h. Equivalently, we show that if y_w is not linked to y_h, then $y_w \notin W \setminus \overline{W}$. Consider the following two cases.

- Node y_w is a root of the closure of C. Since y_w is not linked to y_h, it is not a parent of y_h and from Lemma .17 we have that $y_w \in \overline{W}$.
- Node y_w is not a root of the closure of C. Consider the path from y_h to y_w which has the form $y_h \to \cdots y_{h_1} \to y_w$. Since the node y_{h_1} is hidden, the minimality conditions imply that y_{h_1} has at least another parent, namely y_p (hidden or visible). Since y_p is not a parent of y_h, every rooted subtree $\vec{T}_i$ containing y_p is such that $i \notin I$ (see Lemma .17). Thus, all the visible descendants of y_p (including y_w) are necessarily in $\overline{W}$.

□

C.8 Proof of Theorem 4.7

We first provide the following lemma to ensure that the Steps 24-27 of HCLA correctly merge the fictitious hidden clusters.

Lemma .18. *There exists two distinct nodes y_a, y_b in $W \cap \overline{W}$ such that $y_a, y_b \in \tilde{V}_m$ where $m \notin I$, if and only if y_a and y_b are connected to the same hidden cluster.*

Proof. Observe that $N(C_j^{(i)}) \cup N(C_k^{(i)}) \subseteq W \cap \overline{W}$. Consider two distinct elements y_a, y_b in $W \cap \overline{W}$.

$\Rightarrow$: If we have that $y_a, y_b \in \tilde{V}_m$ where $m \notin I$, then we know that y_a and y_b belong to a commom rooted subtree that does not contain the newly recovered hidden node y_h. Now, by contradiction, assume that the nodes y_a and y_b are not connected to the same hidden cluster. This implies that there exists a path connecting y_a to y_b through y_h. On the other hand, since y_a and y_b belong to a common rooted subtree that does not contain y_h, there exists another path connecting y_a and y_b which does not include y_h. This is a contradiction with the fact that any two nodes in a polytree are connected to each other via at most one path. Therefore, y_a and y_b are connected to the same hidden cluster.

$\Leftarrow$: Let C be the hidden cluster to which both y_a and y_b are connected. Thus, we know that C has y_h as its parent, more specifically, y_h is a parent of one hidden node in C. Let this hidden node be y_{h_1}. Since y_{h_1} has the hidden node y_h as its parent, it is required, by the minimality conditions, that y_{h_1} has at least one other parent. This implies that y_a and y_b are contained in at least one rooted subtree $\vec{T}_m$ which does not contain y_h, namely, $m \notin I$ and also y_a and y_b are contained in $W \cap \overline{W}$. □

Now we can provide the proof of Theorem 4.7.

Proof. HCLA calls the subroutine Hidden Node Detection on all hidden clusters until no more hidden nodes are discovered (Steps 1-3). The goal of Hidden Node Detection is to locate a hidden root y_h in the collapsed quasi-skeleton (V, L, E) of a polytree, determine the visible nodes linked to it and compute the new collapsed quasi-skeleton associated with the visible nodes $V \cup \{y_h\}$. Thus, we just need to show that such subroutine can successfully complete this procedure for a given hidden cluster.

Step 5 simply defines the sets $\tilde{V}_i$ as the visible nodes in the closure of the selected hidden cluster C and Step 6 applies Theorem 4.5 to these sets in order to detect a rooted subtree containing a hidden root of C. Steps 7-14 apply Lemma 4.6 to find all the visible nodes connected to the hidden root of C. If the index set I contains only r, we know that T_r is the only rooted subtree containing y_h and thus $\neg\mathcal{I}(y_h, \emptyset, y)$ for all $y \in V_r$, and $\mathcal{I}(y_h, \emptyset, y)$ for all other visible nodes y. If I contains multiple indices, then Lemma .17 guarantees that $\neg\mathcal{I}(y_h, \emptyset, y)$ for all $y \in W$, and $\mathcal{I}(y_h, \emptyset, y)$ for all other visible nodes y. These last observations are at the core of Steps 15-18.

Observe that the descendants of y_h in the closure of C which are not directly linked to y_h are the nodes in $W \cap \overline{W}$. Steps 19-23 link y_h to these nodes introducing some fictitious hidden clusters. These clusters are just instrumental for the application of the merging algorithm at Step 25. Steps 25-26 merge these fictitious hidden clusters when appropriate as shown in Lemma .18 and they also update the structure of the rooted subtrees containing y_h accordingly. Step 28 merges these hidden clusters using the HCMA considering all the rooted subtrees now that the node y_h can be treated as visible. □

C.9 Proof of Theorem 4.8

Proof. Steps 1-4 are an implementation of the GPT algorithm for the orientation of edges in a polytree. GPT algorithm tests two nodes y_i and y_j on a path of the form $y_i - y_k - y_j$. Thus, all these tests are local in the sense that they are always performed on paths of length 2 in the skeleton of the polytree. However, HRRA performs these tests on paths of length 2 on the quasi-skeleton of the polytree. If the path of length 2 on the quasi-skeleton is the same path of length 2 on the skeleton, HRRA orients the edge the same way the GPT algorithm does. The only difference arises on paths of length 2 in the quasi-skeleton which are not actual paths in the skeleton. This only occurs in situations where a Type-II hidden node is involved on the path.

There are only two possible scenarios when testing the independence statements $\mathcal{I}(y_i, \emptyset, y_j)$ or $\neg\mathcal{I}(y_i, \emptyset, y_j)$ on a path of the form $y_i - y_k - y_j$ in the quasi-skeleton of a minimal latent polytree, as depicted in Figures C.2 and C.3.

The first scenario occurs when we have the path $y_i - y_k - y_j - y_\ell$ on the quasi-skeleton, as in Figure C.2 (a). In this case, there is a yet undetected Type-II hidden node between the nodes y_k and

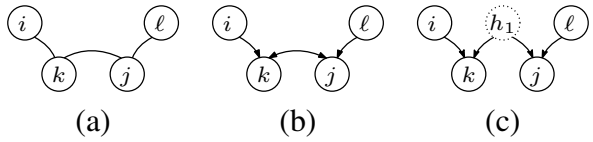


Figure C.2: Quasi-skeleton of a rooted tree with one undiscovered Type-II hidden node (a), the detection of a conflict on the orientation of the edge $y_k - y_j$ (b), and discovery of a Type-II hidden node (c) [4].

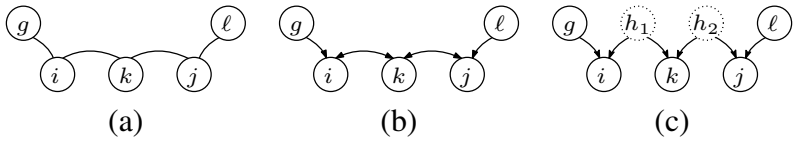


Figure C.3: Quasi-skeleton of a rooted tree with two undiscovered Type-II hidden nodes (a), the detection of a conflict on the orientation of the edges $y_i - y_k$ and $y_k - y_j$ (b), and discovery of two Type-II hidden nodes (c) [4].

y_j, and the node y_ℓ is a parent of the node y_j. In this scenario, we have that $\mathcal{I}(y_i, \emptyset, y_j)$ holds giving the orientations $y_i \rightarrow y_k \leftarrow y_j$. However, because of the Type-II hidden node between the nodes y_k and y_j we also have $\mathcal{I}(y_k, \emptyset, y_\ell)$ implying the orientation $y_k \rightarrow y_j \leftarrow y_\ell$, as in Figure C.2 (b). Thus in this scenario, the presence of the undetected Type-II hidden node is discovered from the double orientation of the edge $y_k - y_j$, as depicted in Figure C.2 (c).

The second scenario occurs when we have the path $y_g - y_i - y_k - y_j - y_\ell$ in the quasi-skeleton, as in Figure C.3 (a). In this case, there are two yet undetected Type-II hidden nodes: one between the nodes y_i and y_k, and one between the nodes y_k and y_j. Following the same reasoning as in the previous scenario, the double orientation of the edges $y_i - y_k$ and $y_k - y_j$ reveals the presence of two Type-II hidden nodes, as depicted in Figures C.3 (b)-(c). □

D Proofs Related to Chapter 5

D.1 Proof of Theorem 5.2

We first provide, for completeness, the proof that the lightest edge in a cut is in every MST of the graph. For definitions of spanning tree, MST, cut and cut edge, please see [76].

Lemma .19 (Cut property)**.** *Let $G = (N, E)$ be a connected undirected graph with a weight function defined on the edges such that $w : E \to \mathbb{R}$. If $\exists\{y_i, y_j\} \in E$ where $y_i \in N_i$ and $y_j \in N_j$ such that $\forall y_k \in N_i$ and $\forall y_\ell \in N_j$ where $\{y_k, y_\ell\} \neq \{y_i, y_j\}$ we have that $w(\{y_i, y_j\}) < w(\{y_k, y_\ell\})$ for some cut $C_G = (N_i, N_j)$, then the edge $\{y_i, y_j\}$ is present in every MST of G.*

Proof. By contradiction, assume that there is a tree T which is an MST of G and does not contain the edge $\{y_i, y_j\}$. Since T is a tree, there exists exactly one path connecting y_i and y_j and therefore adding the edge $\{y_i, y_j\}$ to T creates a cycle. In this cycle, there must be at least one other edge, as well as $\{y_i, y_j\}$, crossing the cut C_G. If we remove this edge, we have a tree which has a total weight lower than the weight of T. This is a contradiction with the assumption that T is an MST. □

Now, we can provide the proof of Theorem 5.2.

Proof. In order to show that SAA is congruous in the skeleton, we need to show that if the LDIM is an LDPT with associated graph $\vec{P}$, then the skeleton is the MST of the fully connected graph with weights of the edges $\{y_i, y_j\}$ equal to $d_L\left(y_i, y_j\right)$. Let Q be the fully connected graph with weights equal to the log-coherence distance values for every pair of nodes. For every edge $\{y_i, y_j\} \in E$, consider the cut $C_Q = (N_i, N_j)$ where N_i is the set of nodes that contains y_i and all the nodes that do not contain y_j on their path to y_i in $\vec{P}$, the associated graph of the LDPT, and $N_j = N \setminus N_i$.

According to Proposition 2.33, the log-coherence distance is additive along the paths of a rooted tree. Also, because of the topological identifiability assumption, we know that the distance of a pair of nodes directly connected with an edge is finite and has a strictly positive value. Therefore, the distance between any node in $N_i \setminus \{y_i\}$ and any node in $N_j \setminus \{y_j\}$ is either infinity or a finite value greater than $d_L\left(y_i, y_j\right)$. Thus, $\{y_i, y_j\}$ will be the edge with the unique minimum weight crossing C_Q and according to Lemma .19, it has to be present in every MST of Q.

Since all the edges in the skeleton of $\vec{P}$ have to be present in every MST of Q and the skeleton of $\vec{P}$ itself is a tree, we conclude that the MST of Q is unique. Therefore, the skeleton of $\vec{P}$ coincides with the unique MST of Q. □

D.2 Proof of Proposition 5.5

Proof. We equivalently show that if we have $\exists (y_i, y_k) \in \vec{E} \setminus \vec{E}_{temp} : (y_k, y_i) \in \vec{E} \setminus \vec{E}_{temp}$, then $A \neq \emptyset$. Since $(y_k, y_i) \in \vec{E} \setminus \vec{E}_{temp}$, according to Step 1 of LOPA, we have that $\exists y_j : (y_j, y_k) \in \vec{E}$ and $d(y_j, y_i) < \infty$. Therefore, we have that $(y_i, y_k) \in \vec{E} \setminus \vec{E}_{temp}$ and $(y_j, y_k) \in \vec{E}$ which imply that $(y_i, y_k, y_j) \in A$, or in other words $A \neq \emptyset$. □

D.3 Proof of Theorem 5.6

Proof. Consider the original LDPT $\mathcal{P}$ and its associated graph $\vec{P} = (N, \vec{E})$ when the distances between the nodes are computed exactly. Let $\bar{P} = (N, E_o, \vec{E}_o)$ be the output of PAA. In this case, $\bar{P}$ will be exactly the same as the output of the GPT algorithm in [68]. Now, we show that these two outputs are exactly the same. Consider Step 3 of ILOAA. In this step, all the inverted forks that are recovered using Corollary 5.3 will be added to $\vec{E}$ since there will not be any conflicts in the orientations of these inverted forks because the original LDIM is an LDPT and the distances are computed exactly. Also, the output of CRLOPA is the same as the output of the propagation step of the GPT algorithm since the original LDIM is an LDPT and the distances are computed exactly.

Since GPT is shown to be congruous in [68] with respect to orientations, then *PAA* is also congruous in orientations. □

Vita

Firoozeh “Dawn” Sepehr holds a BSc degree in Aerospace Engineering and a MSc degree in Space Engineering from Sharif University of Technology, Tehran. She was a graduate research assistant at the University of Tennessee in Knoxville from 2015 to 2019. She obtained her second MSc degree in Computer Science at the University of Tennessee in Knoxville in 2018 and her PhD in Computer Science, under supervision of Dr Donatello Materassi, from the same institution in 2019. Her research interests include graphical models, machine learning, artificial intelligence and graph theory and algorithms.