UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Sede Amministrativa: Università degli Studi di Padova

Dipartimento di Scienze Statistiche
Corso di Dottorato di Ricerca in Scienze Statistiche
Ciclo XXX

Structure Learning of Graphs for Count Data

Coordinatore del Corso: Prof. Nicola Sartori

Supervisore: Prof. Monica Chiogna

Dottoranda: NGUYEN Thi Kim Hue

31 October 2017

Abstract

Biological processes underlying the basic functions of a cell involve complex interactions between genes. From a technical point of view, these interactions can be represented through a graph where genes and their connections are, respectively, nodes and edges.

The main research objective of this thesis is to develop a statistical framework for modelling the interactions between genes when the activity of genes is measured on a discrete scale. We propose several algorithms. First, we define an algorithm for learning the structure of a undirected graph, proving its theoretical consistence in the limit of infinite observations. Next, we tackle structure learning of directed acyclic graphs (DAGs), adopting a model specification proved to guarantee identifiability of the models. Then, we develop new algorithms for both guided and unguided structure learning of DAGs. All proposed algorithms show promising results when applied to simulated data as well as to real data.

Sommario

I processi biologici che regolano le funzioni di base di una cellula sono caratterizzati da numerose interazioni tra geni. Da un punto di vista matematico, è possibile rappresentare queste interazioni attraverso grafi in cui i nodi e gli archi rappresentano, rispettivamente, i geni coinvolti e le loro interazioni.

L'obiettivo principale di questa tesi è quello di sviluppare un approccio statistico alla modellazione delle interazioni tra geni quando questi sono misurati su scala discreta. Vengono a tal fine proposti vari algoritmi. La prima proposta è relativa ad un algoritmo disegnato per stimare la struttura di un grafo non orientato, per il quale si fornisce la dimostrazione di convergenza al crescere delle osservazioni. Altre tre proposte coinvolgono la definizione di algoritmi supervisionati per la stima della struttura di grafi direzionali aciclici, basati su una specificazione del modello statistico che ne garantisce l'identificabilità. Sempre con riferimento ai grafi direzionali aciclici, infine, si propone un algoritmo non supervisionato. Tutti gli algoritmi proposti mostrano risultati promettenti in termini di ricostruzione delle vere strutture quando applicati a dati simulati e dati reali.

Dedication to my family

Acknowledgements

I would like to thank my supervisor, Professor Monica Chiogna, for her patience, encouragement and advice in the past three years. She provided endless support, skilful guidance, and research freedom to me. Her extensive knowledge, enthusiasm have inspired me throughout this PhD project.

I thank my colleagues from Department of Statistical Sciences, University of Padova. I feel incredibly lucky to have shared these three years of ups and downs with such an amazing group of people. I thank Vera for her help and encouragement at various times. I thank Elisa for her help on the analysis of gene sequencing data. I thank my Vietnamese friends for having funny and crazy activities together.

Finally, I thank my family for their constant support and encouragement.

Contents

List of Figures **x**

List of Tables **xii**

Introduction **1**
  Overview 1
  Main contributions of the thesis 3

1 Background **5**
  1.1 Conditional independence and graphs 5
  1.2 Undirected Graphical Models 8
    1.2.1 Separation in undirected graphs 9
    1.2.2 Markov properties on undirected graphs 9
    1.2.3 Factorization 10
  1.3 Directed Acyclic Graphical Models 10
    1.3.1 Factorization 11
    1.3.2 d-connection/separation 11
    1.3.3 Markov properties on directed acyclic graphs 12
    1.3.4 Moralization 12
    1.3.5 Markov equivalent class 12
  1.4 Faithfulness condition 13
  1.5 Background on Structure Learning 14
    1.5.1 Scoring-based Algorithms 14
    1.5.2 Constraint-based Algorithms 15
    1.5.3 Hybrid Algorithms 15

2 Poisson graphical models for count data **17**
  2.1 Model specifications 17
  2.2 Identifiability 19
  2.3 Some Poisson structure learning algorithms 22
    2.3.1 LPGM algorithm 22
    2.3.2 PDN Algorithm 24
    2.3.3 ODS algorithm 25

3 Structure Learning of undirected graphs **27**
  3.1 The PC-LPGM algorithm 27

3.2 Statistical Guarantees 30
3.2.1 Assumptions 31
3.2.2 Consistency of estimators in local models 32
3.2.3 Consistency of the graph estimator 40
3.3 Empirical study 42
3.4 Real data analysis: inferring networks from next generation sequencing data 52
3.5 Discussion 54

4 Guided structure learning of DAGs **57**
4.1 The PK2 algorithm 58
4.1.1 Asymptotic property 59
4.2 The Or-LPGM algorithm 59
4.2.1 Consistency of the Or-LPGM algorithm 61
4.3 The Or-PPGM algorithm 61
4.3.1 Consistency of the Or-PPGM algorithm 62
4.4 Empirical study 63
4.5 Conclusions and remarks 71

5 Unguided structure learning of DAGs **73**
5.1 The learnDAG algorithm 73
5.1.1 Step 1: preliminary neighbourhood selection (PNS) 74
5.1.2 Step 2: estimating parent sets 75
5.1.3 Step 3: pruning of the DAG 78
5.2 Empirical study 79
5.3 Discussion 84

6 Conclusions **87**

Appendix A **89**
A.1 K2 algorithm 89
A.2 PC algorithm 90
A.3 VSL algorithm 92
A.4 GLASSO algorithm 93
A.5 Extension to the nonparanormal model 94
A.6 MMHC algorithm 96

Appendix B **97**
B.1 Appendix B.1 97
B.2 Appendix B.2 103
B.3 Appendix B.3 108

List of Figures

1.1 An example of a conditional independence displayed graphically 6

3.1 The graph structures for $p = 10$ employed in the simulation studies: (a) scale-free; (b) hub; (c) random graph. 44
3.2 The graph structures for $p = 100$ employed in the simulation studies: (a) scale-free; (b) hub; (c) random graph. 44
3.3 Number of TP edges recovered by PC-LPGM; LPGM; PDN; VSL; GLASSO; NPN-Copula; NPN-Skeptic for networks in Figure 3.1 ($p = 10$) and sample sizes $n = 200$, 1000, 2000. First panel row corresponds to high SNR level ($\lambda_{noise} = 0.5$); second panel row corresponds to low SNR level ($\lambda_{noise} = 5$). 46
3.4 PPV (first panel row) and Se (second panel row) for PC-LPGM; LPGM; PDN; VSL; GLASSO; NPN-Copula; NPN-Skeptic for networks in Figure 3.1 ($p = 10$), sample sizes $n = 200$, 1000, 2000 and $\lambda_{noise} = 0.5$. . . . 47
3.5 PPV (first panel row) and Se (second panel row) for PC-LPGM; LPGM; PDN; VSL; GLASSO; NPN-Copula; NPN-Skeptic for networks in Figure 3.1 ($p = 10$), sample sizes $n = 200$, 1000, 2000 and $\lambda_{noise} = 5$. 47
3.6 Number of TP edges recovered by PC-LPGM; LPGM; PDN; VSL; GLASSO; NPN-Copula; NPN-Skeptic for networks in Figure 3.2 ($p = 100$) and sample sizes $n = 200$, 1000, 2000. First panel row corresponds to high SNR level ($\lambda_{noise} = 0.5$); second panel row corresponds to low SNR level ($\lambda_{noise} = 5$). 48
3.7 PPV (first panel row) and Se (second panel row) for PC-LPGM; LPGM; PDN; VSL; GLASSO; NPN-Copula; NPN-Skeptic for networks in Figure 3.2 ($p = 100$), sample sizes $n = 200$, 1000, 2000 and $\lambda_{noise} = 0.5$. . . . 49
3.8 PPV (first panel row) and Se (second panel row) for PC-LPGM; LPGM; PDN; VSL; GLASSO; NPN-Copula; NPN-Skeptic for networks in Figure 3.2 ($p = 100$), sample sizes $n = 200$, 1000, 2000 and $\lambda_{noise} = 5$. 49
3.9 Distribution of four miRNA-Seq: raw data (top), normalized (bottom) data. 53
3.10 Breast cancer miRNA network estimated by the PC-LPGM algorithm (hub nodes coloured red). 54

4.1 The graph structures for $p = 10$ employed in the simulation studies: (a) scale-free; (b) hub; (c) random graph. 64
4.2 The graph structures for $p = 100$ employed in the simulation studies: (a) scale-free; (b) hub; (c) random graph. 65

4.3 Number of TP edges recovered by PKBIC; PKAIC; Or-PPGM; Or-LPGM; PDN; ODS; MMHC; K2mix; K2cut; PCmix; PClog for networks in Figure 4.1 ($p = 10$) and sample sizes $n = 200$, 1000, 2000. 66
4.4 PPV (first panel row) and Se (second panel row) for PKBIC; PKAIC; Or-PPGM; Or-LPGM; PDN; ODS; MMHC; K2mix; K2cut; PCmix; PClog for networks in Figure 4.1 ($p = 10$), sample sizes 200, 1000, 2000. 67
4.5 Number of TP edges recovered by PKBIC; Or-PPGM; Or-LPGM; PDN; ODS; MMHC; K2mix; PClog for networks in Figure 4.2 ($p = 100$) and sample sizes $n = 200$, 1000, 2000. 68
4.6 PPV (first panel row) and Se (second panel row) for PKBIC; Or-PPGM; Or-LPGM; PDN; ODS; MMHC; K2mix; PClog for networks in Figure 4.2 ($p = 100$), sample sizes $n = 200$, 1000, 2000. 69

5.1 An example of applying the PNS step on an undirected graph consisting 6 nodes. 75
5.2 An example of adding edge 3 → 4 based on calculating the score matrix. 77
5.3 Number of TP edges recovered by plearnDAG; olearnDAG; llearnDAG; PDN; ODS; MMHC; PClog for networks in Figure 4.1 ($p = 10$) and sample sizes $n = 200$, 1000, 2000. 80
5.4 PPV (first panel row) and Se (second panel row) for plearnDAG; olearnDAG; llearnDAG; PDN; ODS; MMHC; PClog for networks in Figure 4.1 ($p = 10$), sample sizes $n = 200$, 1000, 2000. 81
5.5 Number of TP edges recovered by plearnDAG; olearnDAG; llearnDAG; PDN; ODS; MMHC; PClog for networks in Figure 4.2 ($p = 100$) and sample sizes $n = 200$, 1000, 2000. 81
5.6 PPV (first panel row) and Se (second panel row) for plearnDAG; olearnDAG; llearnDAG; PDN; ODS; MMHC; PClog for networks in Figure 4.2 ($p = 100$), sample sizes $n = 200$, 1000, 2000. 82

List of Tables

3.1 Monte Carlo marginal means of TP, FP, FN, PPV, Se obtained by simulating 500 samples from each of the three networks shown in Figure 3.1 ($p = 10$). 50

3.2 Monte Carlo marginal means of TP, FP, FN, PPV, Se obtained by simulating 500 samples from each of the three networks shown in Figure 3.2 ($p = 100$). 51

4.1 Monte Carlo marginal means of TP, FP, FN, PPV, Se obtained by simulating 500 samples from each of the three networks shown in Figure 4.1 ($p = 10$). 69

4.2 Monte Carlo marginal means of TP, FP, FN, PPV, Se obtained by simulating 500 samples from each of the three networks shown in Figure 4.2 ($p = 100$). 70

5.1 Monte Carlo marginal means of TP, FP, FN, PPV, Se obtained by simulating 500 samples from each of the three networks shown in Figure 4.1 ($p = 10$). 82

5.2 Monte Carlo marginal means of TP, FP, FN, PPV, Se obtained by simulating 500 samples from each of the three networks shown in Figure 4.2 ($p = 100$). 83

5.3 Monte Carlo marginal means of TP, FP, FN, PPV, Se obtained by simulating 500 samples from random network with edge probability 0.03, $p = 100$. 85

B.1 Simulation results from 500 replicates of the undirected graphs shown in Figure 3.1 for $p = 10$ variables with Poisson node conditional distribution and level of noise $\lambda_{noise} = 0.5$. Monte Carlo means (standard deviations) are shown for TP, FP, FN, PPV and Se. 97

B.2 Simulation results from 500 replicates of the undirected graphs shown in Figure 3.1 for $p = 10$ variables with Poisson node conditional distribution and level of noise $\lambda_{noise} = 5$. Monte Carlo means (standard deviations) are shown for TP, FP, FN, PPV and Se. 99

B.3 Simulation results from 500 replicates of the undirected graphs shown in Figure 3.2 for $p = 100$ variables with Poisson node conditional distribution and level of noise $\lambda_{noise} = 0.5$. Monte Carlo means (standard deviations) are shown for TP, FP, FN, PPV and Se. 100

B.4 Simulation results from 500 replicates of the undirected graphs shown in Figure 3.2 for $p = 100$ variables with Poisson node conditional distribution and level of noise $\lambda_{noise} = 5$. Monte Carlo means (standard deviations) are shown for TP, FP, FN, PPV and Se. 102
B.5 Simulation results from 500 replicates of the DAGs shown in Figure 4.1 for $p = 10$ variables with Poisson node conditional distribution. Monte Carlo means (standard deviations) are shown for TP, FP, FN, PPV and Se. 104
B.6 Simulation results from 500 replicates of the DAGs shown in Figure 4.2 for $p = 100$ variables with Poisson node conditional distribution. Monte Carlo means (standard deviations) are shown for TP, FP, FN, PPV and Se. 106
B.7 Simulation results from 500 replicates of the DAGs shown in Figure 4.1 for $p = 10$ variables with Poisson node conditional distribution. Monte Carlo means (standard deviations) are shown for TP, FP, FN, PPV and Se. 108
B.8 Simulation results from 500 replicates of the DAGs shown in Figure 4.2 for $p = 100$ variables with Poisson node conditional distribution. Monte Carlo means (standard deviations) are shown for TP, FP, FN, PPV and Se. 109

Introduction

Overview

Biological processes in a cell involve complex interactions between genes. These dependencies are commonly represented in the form of a graph. Indeed, graphs are transparent models, easily understood and used by researchers with very different backgrounds. As the direction of influence between genes is a crucial information, directed graphs are often used to represent such networks, a representation also particularly effective when it comes to deal with causal reasoning.

When the problem is to unravel which genes conditionally depend on each other, dependencies have to be inferred from experimental data. Graphical models, which powerfully represent complex multivariate distributions using the adjacency structure of a graph, are widely used to infer gene networks from experimental data. Learning a graphical model from the data boils down, statistically speaking, to making inference on the parameters of the multivariate distribution defined on the set of variables at hand.

The state-of-the-art inference procedures assume that data arise from a multivariate Gaussian distribution. However, high-throughput omics data, such as those from next generation sequencing, often violate this assumption. Here, data are usually discrete, high dimensional, contain a limited number of samples, show a large number of zeros, and come from skewed distributions. A popular choice for adapting to non Gaussian data the large body of results available under the Gaussian assumption relies on data transformation. Indeed, some authors simply apply Gaussian graphical modelling to non Gaussian data after transforming the data by transformations such as log, Box-Cox, copulas, etc. This approach can work well in some circumstances. For example, microarray data are typically Gaussian on a log scale. Unfortunately, they can be also ill-suited, possibly leading to wrong inferences in some circumstances [Gallopin *et al.* (2013)]. This feeds the recent interest for new principled statistical procedures that can deal with different data distributions.

The most natural idea relies on employing a joint multivariate distribution that, beside properly representing the set of relations among the variables, it respects the nature of the variables. A way to achieve this is to incrementally construct multivariate distributions through the specification of conditional distributions. Indeed, the so-called conditional modeling approach, is being recognized as an appealing approach to specify multivariate models that may contain complex dependence structures. Besag (1974) discussed a tractable and natural way to construct multivariate extensions of univariate exponential family distributions. The construction begins with specification of the conditional dependencies present among a finite set of random variables that result in a Markov random field. These conditional dependencies define which of the entries of the multivariate random vector can be considered as neighbours of each other. Various models can then be constructed through specification of local (parametric, semiparametric, nonparametric) regression models.

In this thesis, we assume that conditional distributions follow a Poisson law (extensions to other count distributions could be obtained on the same lines) and we tackle structure learning of both undirected and directed acyclic graphs (DAGs). There are some results in the literature on structure learning of Poisson graphical models for undirected models [Allen and Liu (2013), Yang *et al.* (2013)], or directed graphical models that permit cycles on graphs [Hadiji *et al.* (2015)]. To the best of our knowledge, there is only one algorithm for structure learning of Poisson DAGs, i.e., Park and Raskutti (2015).

The outline of the thesis is as follows. In Chapter 1, we briefly review graphical models and structure learning of graphical models. Poisson graphical models are introduced in Chapter 2. We address both the directed and the undirected framework, proposing a model specification proved to be identifiable. Some Poisson structure learning algorithms are also described. Chapter 3 presents our first proposal, aimed at learning the structure of undirected Poisson graphical models, provides a theoretical analysis of convergence of the algorithm and gives an empirical analysis of its performance. Chapter 4 covers the proposed solutions to guided algorithms. The key ingredient of guided structure learning is assuming that the topological ordering of the set of nodes is specified beforehand. Three new algorithms for refining the graphical structure, i.e., PK2, Or-LPGM, Or-PPGM, are presented along with a theoretical analysis of convergence, and an empirical comparison with a number of different structure learning algorithms. In Chapter 5, we turn our attention to unguided structure learning. This is particularly relevant when the topological ordering may be misspecified, or only a partial ordering on the set of nodes is specified due to a number of reasons. We present the proposed algorithm, and give an empirical analysis of its performance. Chapter 6 contains main conclusions drawn from this project up to date and possible directions for future research.

Main contributions of the thesis

Main contributions of the thesis can be summarized as follows.

1. Definition of a supervised learning algorithm of undirected Poisson graphs, PC-LPGM. The algorithm stems from the local approach of Allen and Liu (2013), where the neighbourhood of each node is estimated in turn by solving a lasso penalized regression problem and the resulting local structures stitched together to form the global graph. To face possible inaccurate inferences when dealing with models of high dimension, we propose to substitute penalized estimation with a testing procedure on the parameters of the local regressions following the lines of the PC algorithm, see Spirtes *et al.* (2000). The PC-LPGM algorithm seems to be very appealing, since it inherits the potential of the PC algorithm that allows to estimate a sparse graph even if the number of nodes, i.e., p, is in the hundreds or thousands.

2. Definition of three partially supervised learning algorithms of DAGs, i.e., PK2, Or-LPGM, Or-PPGM, applicable in situations when some prior information pertaining to the topology of the graph is available. The methods are based on a modification of: a very popular structure learning algorithm, the so-called K2 algorithm [Cooper and Herskovits (1992)]; of the local Poisson graphical model (LPGM) [Allen and Liu (2013)]; of the PC algorithm [Spirtes *et al.* (2000)]. The main strength of these proposals is their simplicity and low computational cost, since exploiting knowledge of the ordering of the nodes allows to considerably reduce the search space.

3. Definition of a supervised learning algorithm of DAGs, learnDAG. We present an algorithm for learning the high dimensional Poisson DAGs, that can be generalized to the case with high variance. The main idea behind our proposal is based on estimating the “potential parent” sets from edge selection in a DAG by using a log likelihood score function. These potential parents are then taken as the input to a pruning step aimed to remove additional edges. Moreover, to make the algorithm feasible to deal with a high dimensional setting, we also include a preliminary neighbourhood selection to reduce dimensionality of the candidate neighbourhood

sets for each node. Results on experimental data show that this algorithm is quite promising in recovering the structure from given data alone.

4. Proof of identifiability of Poisson DAGs. We prove that assuming a Poisson node conditional distributions guarantees identifiability of DAG models.
5. Statistical guarantees (proofs of convergence are provided for most of the proposed algorithms) and extensive empirical studies of performance are given for all proposed algorithms.

Chapter 1

Background

In this chapter, we provide a brief introduction to graphical models. We will address both the directed and the undirected framework, including factorizations of probability distributions, their representations by graphs, and the Markov properties. In detail, we begin with the relation between conditional independence and graphs in Section 1.1. Some basic concepts on undirected graphical models, and directed graphical models are given in Sections 1.2, 1.3 respectively. In Section 1.5, we summarize three approaches of structure learning of graphical models.

1.1 Conditional independence and graphs

The set of conditional independences among a collection of random variables can be intuitively represented as relations defined on the sets of vertices of a graph induced by a certain separation criterion. This connection led Pearl and Paz (1985) to study conditional independence of variables with the help of graphs, where each variable corresponds to a node, and edges of the graph encode the conditional dependences. This application of graphical methods gives rise to the so called graphical models.

In order to give an overview on graphical models, we start by giving the definition of conditional independence.

Definition 1.1.1. We say that random variables X and Y are conditionally independent given the random variable Z and write $X \perp\!\!\!\perp Y|Z$ if and only if

$$\mathbb{P}(X \in A, Y \in B|Z) = \mathbb{P}(X \in A|Z)\mathbb{P}(Y \in B|Z),$$

for any A and B measurable in the sample space of X and Y, respectively.

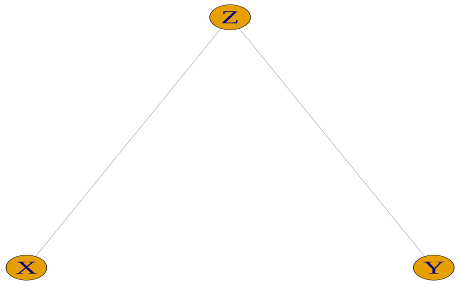


FIGURE 1.1: An example of a conditional independence displayed graphically

Equivalently, we can say that the random variables X and Y are conditionally independent given the random variable Z if and only if $\mathbb{P}(X \in A|Y, Z) = \mathbb{P}(X \in A|Z)$. This alternative definition has an intuitive interpretation: knowing Z renders Y irrelevant for predicting X. It is worth to note that unconditional independence can be seen as a special case of the above definition for Z trivial. However, the two conditions $X \perp\!\!\!\perp Y$, and $X \perp\!\!\!\perp Y|Z$ are very different and will typically not both hold unless we either have $X \perp\!\!\!\perp (Y, Z)$ or $(X, Z) \perp\!\!\!\perp Y$, i.e., if one of the variables is completely independent of both of the others. This fact is a simple form of what is known as Yule-Simpson paradox. For discrete variables, Definition 1.1.1 is equivalent to

$$\mathbb{P}(X = x, Y = y|Z = z) = \mathbb{P}(X = x|Z = z)P(Y = y|Z = z),$$

where the equality holds for all z such that $\mathbb{P}(Z = z) > 0$; whereas for continuous variables the requirement is

$$f(x, y|z) = f(x|z)f(y|z),$$

where the equality holds almost surely. This conditional independence can be displayed graphically as in Figure 1.1. Some fundamental properties of conditional independence are listed bellow.

Proposition 1.1.2. For random variables X, Y, Z, and W it holds

(C1) if $X \perp\!\!\!\perp Y|Z$ then $Y \perp\!\!\!\perp X|Z$;

(C2) if $X \perp\!\!\!\perp Y|Z$ and $U = g(Y)$, then $X \perp\!\!\!\perp U|Z$;

(C3) if $X \perp\!\!\!\perp Y|Z$ and $U = g(Y)$, then $X \perp\!\!\!\perp Y|(Z, U)$;

(C4) if $X \perp\!\!\!\perp Y|Z$ and $X \perp\!\!\!\perp W|(Y,Z)$, then $X \perp\!\!\!\perp (Y,W)|Z$;
if density with respect to (w.r.t.) product measure $f(x,y,z,w) > 0$ and

(C5) if $X \perp\!\!\!\perp Y|(Z,W)$ and $X \perp\!\!\!\perp Z|(Y,W)$ then $X \perp\!\!\!\perp (Y,Z)|W$.

Conditional independence can be seen as encoding abstract irrelevance. With the interpretation: knowing C, A is irrelevant for learning B, then $(C1) - (C4)$ can be translated into:

(I1) if, knowing C, learning A is irrelevant for learning B, then B is irrelevant for learning A;

(I2) if, knowing C, learning A is irrelevant for learning B, then A is irrelevant for learning any part D of B;

(I3) if, knowing C, learning A is irrelevant for learning B, it remains irrelevant having learnt any part D of B;

(I4) if, knowing C, learning A is irrelevant for learning B and, having also learnt A, D remains irrelevant for learning B, then both of A and D are irrelevant for learning B.

The property analogous to $(C5)$ is slightly more subtle and not generally obvious. Moreover, the symmetry property $(C1)$ is a special property of probabilistic conditional independence, rather than that of general irrelevance.

We now recall some important definitions of independence models. Let V be a finite set. An independence model $\perp\!\!\!\perp_\sigma$ over V is a ternary relation over subsets of a finite set V.

Definition 1.1.3 (Semi-graphoid). An independence model is a semi-graphoid if it holds for all subsets A, B, C, D:

(S1) symmetry, if $A \perp\!\!\!\perp_\sigma B|C$ then $B \perp\!\!\!\perp_\sigma A|C$;

(S2) decomposition, if $A \perp\!\!\!\perp_\sigma (B \cup D)|C$ then $A \perp\!\!\!\perp_\sigma B|C$ and $A \perp\!\!\!\perp_\sigma D|C$;

(S3) weak union, if $A \perp\!\!\!\perp_\sigma (B \cup D)|C$ then $A \perp\!\!\!\perp_\sigma B|(C \cup D)$;

(S4) contraction, if $A \perp\!\!\!\perp_\sigma B|C$ and $A \perp\!\!\!\perp_\sigma D|(B \cup C)$, then $A \perp\!\!\!\perp_\sigma (B \cup D)|C$.

Definition 1.1.4 (Graphoid). An independence model is a graphoid if it is a semi-graphoid and satisfies
(S5) intersection, if $A \perp\!\!\!\perp_\sigma B|(C \cup D)$ and $A \perp\!\!\!\perp_\sigma C|(B \cup D)$ then $A \perp\!\!\!\perp_\sigma (B \cup C)|D$.

Definition 1.1.5 (Compositional). An independence model is compositional if it satisfies (S6) composition, if $A \perp\!\!\!\perp_\sigma B|C$ and $A \perp\!\!\!\perp_\sigma D|C$ then $A \perp\!\!\!\perp_\sigma (B \cup D)|C$.

Note 1.1.6. The composition property ensures that pairwise conditional independence implies setwise conditional independence, i.e.,

$$A \perp\!\!\!\perp_\sigma B|C \Leftrightarrow \alpha \perp\!\!\!\perp_\sigma \beta|C, \ \forall \alpha \in A, \beta \in B.$$

For a system V of labelled random variables $X_v, v \in V$, with distribution P we can then define an independence model $\perp\!\!\!\perp_P$ by

$$A \perp\!\!\!\perp B|C \Leftrightarrow \mathbf{X}_A \perp\!\!\!\perp_P \mathbf{X}_B|\mathbf{X}_C,$$

where $\mathbf{X}_A = (X_v, v \in A)$ denotes the variables with labels in A. Obviously, general properties of conditional independence $(C1) - (C4)$ imply that this independence model satisfies the semi-graphoid axioms, and the graphoid axioms if the joint density of the variables is strictly positive. An independence model of this kind is said to be probabilistic. We note that a probabilistic independence model is not compositional in general, however when P is a regular multivariate Gaussian distribution, then it is a compositional graphoid.

1.2 Undirected Graphical Models

In what follows, we consider a p-dimensional random vector $\mathbf{X} = (X_1, \dots, X_p)$ such that (s.t.) each random variable X_s corresponds to a node of the graph $G = (V, E)$ with index set $V = \{1, 2, \dots, p\}$. In an undirected graph $G = (V, E)$, an edge between two nodes s and t, is denoted by (s, t). The neighbourhood of node $s \in V$ is defined to be the set $N(s)$ consisting of all nodes connected to s, i.e., $N(s) = \{t \in V : \quad (s, t) \in E\}$. A path in G is a sequence of (at least two) distinct nodes $j_1, \dots, j_n$ s.t. there is an edge between j_k and j_{k+1} , for all $k \in \{1, 2, \dots, n\}$. A cycle is a path with the modification that $j_1 = j_n$. If $A \subset V$ is a subset of a vertex set it induces a subgraph $G_A = (A, E_A)$, where $E_A \subset E$ is obtained from E so that only edges with both endpoints in A are kept. A graph is complete if all its nodes are joined by an edge. A subset is complete if it induces a complete subgraph. A complete set that is maximal is called a clique.

1.2.1 Separation in undirected graphs

Here, we recall an important concept of undirected graphs that represents the conditional independence of random variables, i.e., separation.

Definition 1.2.1 (Separation). Let $G = (V, E)$ be finite and simple undirected graph (no self-loops, no multiple edges). For subsets A, B, S of V, we say that S separates A from B in G, and denote $A \perp\!\!\!\perp_G B|S$ if all paths from A to B intersect S.

Note 1.2.2. It is easy to see that the separation relation on subsets of V is a compositional graphoid; and the name "graphoid" for such separation relations also comes from this reason.

1.2.2 Markov properties on undirected graphs

We now review briefly three Markov properties associated with undirected graphs, see Lauritzen (1996) for a detailed presentation. Let $\mathbf{X}_U = (X_j : j \in U \subset V)$ be the random vector of all variables in $U \subset V$. The random vector $\mathbf{X}$ satisfies the pairwise Markov property w.r.t. G if

$$X_s \perp\!\!\!\perp X_t | \mathbf{x}_{V \setminus \{s,t\}},$$

whenever $(s, t) \notin E$. Moreover, $\mathbf{X}$ satisfies the local Markov property w.r.t. G if

$$X_s \perp\!\!\!\perp \mathbf{X}_{V \setminus \{N(s) \cup \{s\}\}} | \mathbf{x}_{N(s)},$$

for every node $s \in V$. Finally, $\mathbf{X}$ satisfies the global Markov property w.r.t. G if

$$\mathbf{X}_A \perp\!\!\!\perp \mathbf{X}_B | \mathbf{x}_C,$$

for any triple of pairwise disjoint subsets $A, B, C \subset V$ such that C separates A and B in G, i.e., every path between a node in A and a node in B contains a node in C.

Note 1.2.3. For any semigraphoid it holds that the global Markov property implies the local Markov property, which in turn implies the pairwise Markov property. Although not true in general, the three Markov properties are equivalent when the independence relation on G satisfies graphoid axioms, that happens for example, when all variables $X_s, s \in V$, are discrete with positive joint probabilities, or when $\mathbf{X}$ has a positive and continuous density with respect to Lebesgue measure.

In an undirected graphical model, the pairwise Markov property infers a collection of full conditional independences encoded in absent edges. For this reason, performing $\binom{|V|}{2}$ pairwise full conditional independence tests yields a method to estimate the graph

G. The local Markov property means that every variable is conditionally independent of the remaining ones, given its neighbours. Hence, this property suggests that each variable $X_s,\ s \in V$, can be optimally predicted from its neighbour $\mathbf{X}_{N(s)}$, giving rise to the so called neighbourhood selection method. Under the global Markov property, one should look for separating sets to find conditional independence relations.

1.2.3 Factorization

Graphical models can also be defined in terms of density factorizations, as the joint distribution can be expressed as a product of clique-wise compatibility functions. Consider a graph $G = (V, E)$ with vertex set $V = \{1, \dots, p\}$ corresponding to p variables and edge set $E \subset V \times V$. Then, the definition of factorization is given as follows.

Definition 1.2.4 (Factorization). Suppose $\mathbf{X} = (X_1, \dots, X_p)$ has a density w.r.t. a product measure $\mu = \otimes_{s=1}^{p} \mu_s$ on $\mathbb{R}^{|V|}$, where μ_s is usually a Lebesgue measure if X_s is a continuous random variable, or counting measure if X_s is discrete. Let $\mathcal{C}(G)$ be the set of all complete subsets (cliques) of G, i.e., $C \in \mathcal{C}(G)$ if $(s, t) \in E$ for all $s, t \in C$. Then, the distribution of $\mathbf{X}$ is said to factorize according to G if it has the density of the form

$$f(\mathbf{x}) = \prod_{C \in \mathcal{G}} \phi_C(\mathbf{x}_C),$$

where the potential function $\phi_C(.)$ is a function of $|C|$ variables. Thus, a graphical model can be defined by specifying families of potential functions.

Note 1.2.5. If $f(.)$ factorizes, then it satisfies the global Markov property, as well as all weaker Markov properties (Hammersley and Clifford theorem), and, in the case of a graphoid, all the properties coincide.

1.3 Directed Acyclic Graphical Models

Consider a p-dimensional random vector $\mathbf{X} = (X_1, \dots, X_p)$ with index set $V = \{1, 2, \dots, p\}$. A directed graph $\mathcal{G} = (V, E)$ is a structure consisting of a finite set V of vertices (also called nodes) and a finite set E of directed edges (also called arcs) between these vertices. A directed edge from node k to node j is denoted by (k, j) or $k \to j$, k is called a parent of node j, and j is a child of k. The set of parents of a vertex j, denoted $pa(j)$, consists of all parents of node j, similarly for the set of children $ch(j)$. If there is a directed path from k to j, then k is an ancestor of j and j is a descendant of k. The ancestors of j are $an(j)$ and the descendants are $de(k)$; similarly for sets of nodes A we use $an(A)$, and $de(A)$. The non-descendants of a node k are $nd(k) = V \backslash (\{k\} \cup de(k))$. Two nodes that are connected by an edge are called adjacent. A triple of nodes (i, j, k) is an unshielded triple if i and j are adjacent to k but i and j are not adjacent. An unshielded triple (i, j, k) forms a v-structure if $i \rightarrow k$ and $j \rightarrow k$. In this case k is called a collider. The skeleton of a graph G is the set of all edges in G without direction, and denoted by $ske(G)$.

Definition 1.3.1 (Partially directed acyclic graph). A graph is said to be a partially directed acyclic graph (PDAG) if there is no directed cycle, i.e., there is no pair (j, k) such that there are directed paths from j to k and from k to j.

Definition 1.3.2 (Directed acyclic graph). A directed acyclic graph (DAG) is a PDAG with all directed edges. In DAGs, a topological ordering $j_1, \ldots, j_p$ is an order of p nodes such that there are no directed paths from j_k to j_t if $k > t$ for all $k,\ t \in V$.

Definition 1.3.3 (Topological ordering). A topological ordering of vertices of a directed acyclic graph is such that if a variable X is an ancestor of a variable Y in a graph G, then X precedes Y in that ordering.

Note 1.3.4. Obviously, such an ordering is generally non unique, but always exists.

1.3.1 Factorization

Using the locality defined by the parent-child relationship, the joint probability distribution can be factorized as the product of conditional densities.

Definition 1.3.5 (Factorization). Suppose $\mathbf{X}$ has a density with respect to a product measure. Then, the distribution of $\mathbf{X}$ factorizes according to a DAG $G = (V, E)$ if it has a density of the form

$$f(x) = \prod_{j \in V} f(x_j | \mathbf{x}_{pa(j)}),$$

where the $f(x_j | \mathbf{x}_{pa(j)})$ are conditional densities with $f(x_j | \mathbf{x}_{\emptyset}) = f(x_j)$.

Note 1.3.6. We note that directed graphical models with directed cycles may not satisfy the factorization property.

1.3.2 d-connection/separation

In a DAG, independence is encoded by the relation of d-separation, defined below.

Definition 1.3.7 (d-connection/separation). Two nodes j and k in V are d-connected given $S \subset V \backslash \{j, k\}$ if G contains a path π with endpoints j and k s.t.

(i) for every collider v on π, either v or a descendent of v is in S, and

(ii) no non-collider on π is in S.

Generalizing to sets, two disjoint subsets $A, B \subset V$ are d-connected given $S \subset V \backslash (A \cup B)$ if there are two nodes $j \in A$ and $k \in B$ that are d-connected given S. If this is not the case, then S d-separates A and B.

1.3.3 Markov properties on directed acyclic graphs

Similarly to what we have done for undirected graphs, we state the counterparts of Markov properties in DAGs.

A random vector $\mathbf{X} = (X_v : v \in V)$ satisfies the local Markov property w.r.t. a DAG G if

$$X_v \perp\!\!\!\perp \mathbf{X}_{nd(v)\backslash pa(v)} | \mathbf{X}_{pa(v)},$$

for every $v \in V$. Similarly, $\mathbf{X}$ satisfies the global Markov property w.r.t. G if

$$\mathbf{X}_A \perp\!\!\!\perp \mathbf{X}_B | \mathbf{X}_C$$

for all triples of pairwise disjoint subsets $A, B, C \subset V$ s.t. C d-separates A and B in G, which we denote by $A \perp\!\!\!\perp B | C$.

Note 1.3.8. It is always true for a DAG that the global property, local Markov property, and the factorization property are equivalent [Lauritzen *et al.* (1990)].

1.3.4 Moralization

A moral graph is a concept in graph theory, used to find the equivalent undirected form of a DAG.

Definition 1.3.9 (Moralization). The moral graph G_m of a DAG G is obtained by adding undirected edges between unmarried parents and subsequently dropping directions.

Note 1.3.10. If P factorizes w.r.t. G, then it factorizes w.r.t. the moralised graph G_m. Hence, if P satisfies any of the directed Markov properties w.r.t. G, then it satisfies all Markov properties for G_m. Moreover, it holds that $A \perp\!\!\!\perp_G B | S$ if and only if S separates A from B in this undirected graph G_m [Lauritzen (1996)].

1.3.5 Markov equivalent class

Every DAG determines a set of conditional independence relations among variables. It turns out that different directed graphs entail the same d-separation rules. These graphs are Markov equivalent and the set of Markov equivalent graphs is called a Markov equivalence class. The formal definition of Markov equivalence class is as follows:

Definition 1.3.11 (Markov equivalence). Two graphs G_1 and G_2 are Markov equivalent if their independence models coincide, i.e., if $A \perp\!\!\!\perp_{G_1} B|S \Leftrightarrow A \perp\!\!\!\perp_{G_2} B|S$

Note 1.3.12. Two directed acyclic graphs G_1 and G_2 are Markov equivalent if and only if they have the same skeleton $ske(G_1) = ske(G_2)$ and the same unshielded collider tripaths.

The non uniqueness of the graphical representation has important practical consequences in statistical inference: in the problem of inferring the graphical structure from data, the underlying DAG is not identifiable, i.e., we cannot distinguish between DAGs in the same equivalence class. Hence, structure learning algorithms usually output an object representative of the whole equivalence class, the so called complete partially directed acyclic graph (CPDAG). A CPDAG is a graph that contains both directed and undirected edges. An edge between nodes i and j is directed in a CPDAG if and only if the orientation of that edge is the same across all DAGs in the equivalence class, otherwise we keep it undirected.

1.4 Faithfulness condition

The connection between graphs and probability distributions established by the concept of Markov property has been presented in Section 1.2 and Section 1.3 for undirected graphs and DAGs respectively. In particular, if there is a specific type of separation between nodes i and j of the graph given the node subset C then random variables X_i and X_j are conditionally independent given the random vector $\mathbf{X}_C$ in the probability distribution. However, the "ultimate" connection between probability distributions and graphs requires the other direction to hold, namely for every conditional independence in the probability distribution to correspond to a separation in the graph. This connection has been called faithfulness of the probability distribution [Sadeghi (2017)].

Definition 1.4.1 (Faithfulness). A distribution $\mathbb{P}$ is faithful to a graph G if no conditional independence relations other than the ones entailed by the Markov property are presented.

Note 1.4.2. Let $J(G)$ be the independence model induced by the graph G, and $J(\mathbb{P})$ be the independence model induced by the distribution $\mathbb{P}$. Then, we say that $\mathbb{P}$ is faithful to G if $J(\mathbb{P}) = J(G)$.

The faithfulness condition holds in some cases, such as, for example, the Gaussian or the discrete case [Meek (1995)]. However, it does not hold in general.

1.5 Background on Structure Learning

When talking about learning in the graphical modelling framework, we distinguish two broad classes of problems: given a structure and a model specification, estimation of the model; and inferring the structure of the model from observation data. While the former is usually considered a traditional problem of statistical inference, the latter is usually covered in the machine learning literature. Here, we consider the latter problem, the so-called structure learning. In detail, given a set of observations that can be assumed to be independent and identically distributed (i.i.d.) instances sampled from a probability distribution $\mathbb{P}$ corresponding to a graph G, we aim to recover the structure of the graph G.

Structure learning algorithms can be roughly divided into two major approaches: scoring-based algorithms and constraint-based algorithms. Some hybrid algorithms are often defined mixing the two previous classes. The three approaches are touched upon in what follows. A description of some algorithms for continuous and categorical data is also provided in Appendix A, with special focus on the algorithms which we have used in our empirical studies.

1.5.1 Scoring-based Algorithms

Scoring-based algorithms make use of (i) a score function, such as, for example, the BIC score [Schwarz (1978)], the AIC score [Akaike (1974)], the Bayesian Dirichlet score [Heckerman *et al.* (1995)], the likelihood-equivalence Bayesian Dirichlet score [Heckerman *et al.* (1995)], indicating how well the network fits the data; and (ii) a search strategy, which searches the model that maximizes the score in a possible structure space. We recall here the concept of consistency of a scoring criterion.

Definition 1.5.1 (Consistency of a scoring criterion)*.* Assume that data are generated by some distribution $\mathbb{P}^*$ whose underlying DAG is G^* (in other words, the set of conditional independence relations that hold in $\mathbb{P}^*$ coincides with the set of conditional independence relations implied by G^*). We say that scoring function is consistent if the following properties hold as the number of observations goes to infinity, with probability that approaches 1:

i) the structure G^* will maximize the score;

ii) all structures that are not equivalent to G^* will have strictly lower score.

Especially when dealing with DAGs, the number of possible structures grows exponentially as a function of the number of nodes. For this reason, they often employ a greedy search, since searching over the space of possible networks is NP-hard. Chickering and Meek (2002) derived optimality results stating that the greedy search used in conjunction with any consistent scoring criterion will, as the number of observations goes to infinity, identify the true structure (up to an equivalence class).

1.5.2 Constraint-based Algorithms

Constraint-based algorithms make use of some statistical tests, such as, for example, the χ^2 test of independence in contingency tables [Spirtes *et al.* (2000)], or the test on partial correlation under the Gaussian assumption [Kalisch and Bühlmann (2007)] to find conditional independence relations among the variables. When dealing with DAGs, an additional task needs to be considered, i.e., identifying the direction of edges on the graph. Here, the results of the conditional independent tests are used in conjunction with orientation rules to construct DAGs. This approach, in general, does not entail a unique graph when learning DAGs. As a consequence, results of constraint-based algorithms are often in the form of a CPDAG, i.e., an object representative of the whole equivalence class.

1.5.3 Hybrid Algorithms

Another class of methods, called hybrid methods, takes ideas from both the above described approaches. This approach is typically designed for learning DAGs. It usually works in two steps: firstly, conditional independence tests of the constraint based methods are used to learn the skeleton of a the network; then, search-and-score methods are performed over the graph structure space that respects the skeleton output to find the optimal structure. The resulting algorithms inherit advantages of both approaches. However, results of hybrid algorithms also suffer from disadvantages of both constraint-based algorithms and score-based algorithms, where algorithms require strong assumptions and identify a graph up to Markov equivalence class.

Chapter 2

Poisson graphical models for count data

In this chapter, we focus on Poisson graphical models. We address both the directed and the undirected framework, providing model specifications. As identifiability is an important issue of learning DAGs, we tackle identifiability of Poisson graphical models, proving that the proposed model specification leads to identifiable models. Some structure learning algorithms are also described. In detail, Poisson model specification is given in Section 2.1. We prove that models are identifiable in Section 2.2. The Possion structure learning algorithms are reviewed in Section 2.3.

2.1 Model specifications

We will take a conditional approach as the starting point to model specification. In detail, assume that each conditional distribution of node X_s given a conditional set $\mathbf{K}$ follows a Poisson distribution, i.e.,

$$X_s | \mathbf{x_K} \sim \mathrm{Pois}(f_s(\mathbf{x_K})), \tag{2.1}$$

where $\mathbf{K}$ is the set of neighbours of s, -denoted as $N(s)$ in Poisson undirected graphs, -or the set of parents of s, -denoted as $pa(s)$ in Poisson DAGs. The function $f_s(.)$ is an arbitrary function representing the mean of the conditional distribution.

Different choices of the function $f_s(.)$ give rise to different model specifications. The most usual choice for $f_s(.)$ is the link function of the univariate Poisson generalized linear models, i.e.,

$$f_s(\mathbf{x_K}, \boldsymbol{\theta}_s) = \exp\{\theta_s + \sum_{t \in \mathbf{K}} \theta_{st} x_t\} = \exp\{\theta_s + \sum_{t \neq s} \theta_{st} x_t\}, \qquad (2.2)$$

where $\theta_{st} = 0$ if $t \notin \mathbf{K}$, and $\boldsymbol{\theta}_s = \{\theta_s, \theta_{st},\ s \in V, t \in V \backslash \{s\}\}$. Then, the node conditional distribution is

$$\mathbb{P}_{\boldsymbol{\theta}_s}(x_s | \mathbf{x_K}) \;=\; \exp\big\{\theta_s x_s + \sum_{t \in \mathbf{K}} \theta_{st} x_s x_t - \log(x_s!) - e^{\theta_s + \sum_{t \in \mathbf{K}} \theta_{st} x_t}\big\},$$

where $\boldsymbol{\theta}_s \in \boldsymbol{\Theta}_s \subset \mathbb{R}^p$. This specification defines an undirected graph $G = (V, E)$ in which one edge between node s and node t implies $\theta_{st} \neq 0$ and $\theta_{ts} \neq 0$ (or $f_s(.)$ depends on x_t and $f_t(.)$ depends on x_s). A missing edge between node s and node t corresponds to the condition $\theta_{st} = 0$ or $\theta_{ts} = 0$ (or $f_s(.)$ does not depend on x_t or $f_t(.)$ does not depend on x_s), implying conditional independence of X_s and X_t given the neighbours $\mathbf{X_K}$, i.e., $X_s \perp\!\!\!\perp X_t | \mathbf{x_K}$.

Similarly, in case of DAGs, the above specification puts an edge from node s to node t if $\theta_{st} \neq 0$ (or $f_s(.)$ depends on x_t). A missing edge $s \to t$ corresponds to the condition $\theta_{st} = 0$ (or $f_s(.)$ does not depend on x_t), implying conditional independence of X_s and X_t given the parents of s, i.e., $X_s \perp\!\!\!\perp X_t | \mathbf{x}_{pa(s)}$.

For DAGs, the joint distribution is easily obtained, since the joint distribution factorizes in terms of conditional distributions, leading to

$$\begin{aligned} \mathbb{P}_{\boldsymbol{\theta}}(\mathbf{x}) &= \prod_{s=1}^{p} \mathbb{P}_{\boldsymbol{\theta}_s}(x_s | \mathbf{x}_{pa(s)}) \qquad (2.3) \\ &= \exp\big\{\sum_{s=1}^{p} \theta_s x_s + \sum_{(s,t) \in E} \theta_{st} x_s x_t - \sum_{s=1}^{p} \log(x_s!) - \sum_{s=1}^{p} e^{\theta_s + \sum_{t \in pa(s)} \theta_{st} x_t}\big\}. \end{aligned}$$

where $\boldsymbol{\theta} = \{\boldsymbol{\theta_s},\ \boldsymbol{s \in V}\} \in \boldsymbol{\Theta} \subset \mathbb{R}^{p \times p}$. In case of undirected structures, if (and only if) $\theta_{st} \leq 0 \ \ \forall\ s,\ t$, the joint distribution over all random variables can be expressed in the following symbolic form,

$$\mathbb{P}_{\boldsymbol{\theta}}(\mathbf{x}) = \exp\big\{\sum_{s=1}^{p} \theta_s x_s + \sum_{s=1}^{p} \sum_{t=1}^{p} \theta_{st} x_s x_t - \sum_{s=1}^{p} \log(x_s!) - A(\boldsymbol{\theta})\big\}, \qquad (2.4)$$

where $A(\boldsymbol{\theta})$ is a normalizing constant, see Allen and Liu (2013). In other words, a unique consistent joint distribution exists provided that conditional dependencies are all negative, a condition also known as "competitive relationship" among variables, which highly limits use of the distribution in applications. To overcome the above mentioned limitation, some authors suggest to modify base measures (or sufficient statistics) so as to guarantee the existence of a joint distribution allowing a richer dependence structure. The approach gives rise to the so called Truncated Poisson Graphical Models (TPGM), Quadratic Poisson Graphical Models (QPGM), and Sub-linear Poisson Graphical Models (SPGM) [Yang *et al.* (2013)]. These three classes of graphical models permit rich dependence structures between variables, although they still have certain limitations on types of dependencies, or have a Gaussian-esque thin tail, or allow only a bounded range data.

But the question is if theoretical existence of a consistent joint distribution is required in real life applications, in particular when the interest is the structure of dependences. Indeed, some authors suggest to consider local conditional probability models regardless of the existence of the joint distribution. In these cases, various models can be constructed through specification of local (parametric, semiparametric, nonparametric) regression models. These, combined with clever estimation solutions, give rise to a wide set of solutions able to embrace an extremely wide range of real situations. For example, to face sparsity of the graph, penalized regression techniques might be applied to estimate each conditional regression (see Allen and Liu (2013), Gallopin *et al.* (2013), Žitnik and Zupan (2015)). A class of semiparametric models is proposed in Yang *et al.* (2014) along with an adaptive multistage convex relaxation algorithm to estimate parameters in the model. The conditional mean could also be an arbitrary function, as in Hadiji *et al.* (2015).

2.2 Identifiability

An important issue of learning DAGs is the identifiability of the models. As stated before, different DAGs can encode the same set of conditional independences. Therefore, DAG models can be defined only up to their Markov equivalence class. However, in some cases, it is possible to identify the DAG by exploiting specific properties of the distribution, see Peters and Bühlmann (2013), Shimizu *et al.* (2006), Peters *et al.* (2012). Here, we prove that the Poisson DAG models in (2.3) are also identifiable.

It is worth remembering that an alternative proof of the identifiability of Poisson DAG models was also given in Park and Raskutti (2015). The Authors proved their result assuming a condition (Theorem 2.3.2, Section 2.3.3) which is in fact redundant in the setting under consideration. For this reason, we report here a detailed proof which benefits from the ideas developed in the work of Peters and Bühlmann (2013).

Consider a p-random vector $\mathbf{X} = (X_1, \ldots, X_p)$, and assume that the node conditional distributions follow a Poisson distribution, i.e.,

$$X_s|\mathbf{x}_{pa(s)} \sim \text{Pois}(\exp\{\theta_s + \sum_{t\in pa(s)} \theta_{st}x_t\}), \quad s \in V = \{1, \ldots, p\}. \tag{2.5}$$

For a fixed graph G, let $pa_G(j)$, $ch_G(j)$, and $nd_G(j)$ be the set of parents, the set of children, and the set of non-descendants of node j in G, respectively.

Proposition 2.2.1. Let $\mathbf{X}$ be a p-random vector defined as in (2.5) and $G = (V, E)$ be a DAG. Consider a variable $X_j,\ j \in V$, and one of its parents $k \in pa_G(j)$. For all set S with $pa_G(j)\backslash\{k\} \subseteq S \subseteq nd_G(j)\backslash\{k\}$, we have $X_j \not\perp\!\!\!\perp X_k|\mathbf{X}_S$.

Proof. This proposition can be proved easily by using the definition of d-connection and the faithfulness assumption. Indeed, for a fixed node $j \in V$, for all $k \in pa_G(j)$ and for all set S satisfies $pa_G(j)\backslash\{k\} \subseteq S \subseteq nd_G(j)\backslash\{k\}$, there always exists the path $k \to j$ satisfies the definition of d-connection. Hence, $X_j \not\perp\!\!\!\perp X_k|\mathbf{X}_S$. □

Theorem 2.2.2. The Poisson DAG defined as in (2.5) is identifiable.

Proof. Assume there are two structure models as in (2.5) which both encode the same set of conditional independences, one with graph G, and the other with graph G'. We will show that $G = G'$.

Since DAGs do not contain any cycles, we can always find one node without any child. Indeed, assume to start at some node, and follow a directed path that contains the chosen node. After at most $\#(V-1)$ steps, a node without any child is reached. Eliminating such a node from the graph leads to a new DAG.

We repeat this process on G and G' for all nodes that have: (i) no children, (ii) the same parents in G and G'. This process terminates with one of two possible outputs: (a) no nodes left; (b) a subset of variables, which we call again $\mathbf{X}$, two sub-graphs, which we call again G and G', and a node j that has no children in G s.t. either $pa_G(j) \neq pa_{G'}(j)$ or $ch_{G'}(j) \neq \emptyset$. If (a) occurs, the two graphs are identical and the result is proved. In what follows, we consider the case that (b) occurs.

For such a j node, we have

$$X_j \perp\!\!\!\perp X_{V\backslash(pa_G(j)\cup\{j\})}|\mathbf{X}_{pa_G(j)}, \tag{2.6}$$

thanks to the Markov properties with respect to G. To make our argument clear, we divide the set of parents $pa_G(j)$ into three disjoint partitions W, Y, Z representing, respectively, the set of common parents in both graphs; the set of parents in G being a subset of children in G'; the set of parents in G which are not parents in G'. Formalizing,

- $Z = pa_G(j) \cap pa_{G'}(j)$;
- $Y \subset pa_G(j)$ s.t. $ch_{G'}(j) = Y \cup T$;
- $W \subset pa_G(j)$ s.t. W are not adjacent to j in G'.

Thus,

$$\begin{aligned} pa_G(j) &= W \cup Y \cup Z, \qquad ch_G(j) = \emptyset, \\ pa_{G'}(j) &= D \cup Z, \qquad ch_{G'}(j) = T \cup Y, \end{aligned}$$

where D is not adjacent to j in G. Let $U = W \cup Y$ and consider the following two cases:

- $U = \emptyset$. Then, there exists a node $d \in D$ or a node $t \in T$, otherwise j would have been discarded.
  - If there exists a node $d \in D$, (2.6) implies $X_j \perp\!\!\!\perp X_d | \mathbf{X}_Q$, for $Q = Z \cup D \backslash \{d\}$, which contradicts Proposition (2.2.1) applied to G'.
  - If $D = \{\emptyset\}$, and there exists a node $t \in T$, then (2.6) implies $X_j \perp\!\!\!\perp X_t | \mathbf{X}_Q$, for $Q = Z \cup pa_{G'}(t) \backslash \{j\}$, which contradicts Proposition (2.2.1) applied to G'.
- $U \neq \emptyset$. We note that, within the structure of the graph G', the Poisson assumption implies
  $$\mathrm{Var}\big(X_j | \mathbf{X}_{pa_{G'}(j)}\big) = \mathbb{E}\big(X_j | \mathbf{X}_{pa_{G'}(j)}\big). \tag{2.7}$$
  However, by applying the law of total variance we get
  $$\begin{aligned} \mathrm{Var}\big(X_j | \mathbf{X}_{pa_{G'}(j)}\big) \;&=\; \mathrm{Var}\big(\mathbb{E}(X_j | \mathbf{X}_{pa_{G'}(j)} \cup \mathbf{X}_{pa_G(j)}) | \mathbf{X}_{pa_{G'}(j)}\big) \\ &\quad + \mathbb{E}\big(\mathrm{Var}(X_j | \mathbf{X}_{pa_{G'}(j)} \cup \mathbf{X}_{pa_G(j)}) | \mathbf{X}_{pa_{G'}(j)}\big). \end{aligned}$$
  By applying Property (2.6) we can rewrite
  $$\mathrm{Var}\big(X_j | \mathbf{X}_{pa_{G'}(j)}\big) = \mathrm{Var}\big(\mathbb{E}(X_j | \mathbf{X}_{pa_G(j)}) | \mathbf{X}_{pa_{G'}(j)}\big) + \mathbb{E}\big(\mathrm{Var}(X_j | \mathbf{X}_{pa_G(j)}) | \mathbf{X}_{pa_{G'}(j)}\big). \tag{2.8}$$
  In graph G, we have $X_j | \mathbf{X}_{pa_G(j)} \sim \mathrm{Pois}(f_j(\mathbf{X}_{pa_G(j)}))$, so that
  $$\mathbb{E}(X_j | \mathbf{X}_{pa_G(j)}) = \mathrm{Var}(X_j | \mathbf{X}_{pa_G(j)}) = f_j(\mathbf{X}_{pa_G(j)}).$$

Hence, from Equation (2.8), we get

$$\begin{aligned}\mathrm{Var}\big(X_j|\mathbf{X}_{pa_{G'}(j)}\big) &= \mathrm{Var}\big(f_j(\mathbf{X}_{pa_G(j)})|\mathbf{X}_{pa_{G'}(j)}\big) + \mathbb{E}\big(\mathbb{E}(X_j|\mathbf{X}_{pa_G(j)})|\mathbf{X}_{pa_{G'}(j)}\big) \qquad (2.9)\\ &= \mathrm{Var}\big(f_j(\mathbf{X}_{pa_G(j)})|\mathbf{X}_{pa_{G'}(j)}\big) + \mathbb{E}\big(\mathbb{E}(X_j|\mathbf{X}_{pa_G(j)} \cup \mathbf{X}_{pa_{G'}(j)})|\mathbf{X}_{pa_{G'}(j)}\big)\\ &= \mathrm{Var}\big(f_j(\mathbf{X}_{pa_G(j)})|\mathbf{X}_{pa_{G'}(j)}\big) + \mathbb{E}\big(X_j|\mathbf{X}_{pa_{G'}(j)}\big),\end{aligned}$$

by applying (2.6). Equation (2.9) implies

$$\mathrm{Var}\big(X_j|\mathbf{X}_{pa_{G'}(j)}\big) > \mathbb{E}\big(X_j|\mathbf{X}_{pa_{G'}(j)}\big),$$

since $\mathrm{Var}\big(f_j(\mathbf{X}_{pa_G(j)})|\mathbf{X}_{pa_{G'}(j)}\big) > 0$ in general, except at the root node.

□

2.3 Some Poisson structure learning algorithms

In this section, we briefly review three structure learning algorithms which make use of the Poisson assumption.

2.3.1 LPGM algorithm

The local Poisson graphical model (LPGM) algorithm is a scoring-based algorithm, designed to recover undirected Poisson graphical models. Introduced by Allen and Liu (2013), it assumes that the node conditional distributions follow a Poisson law, where the function $f_j(.)$ is taken to be the link function of the univariate Poisson generalized linear models. It works by locally fitting a l_1- penalized log-linear regression to each node, i.e., to each random variable X_s, given all other variables $\mathbf{X}_{V\setminus\{s\}}$ as predictors. The estimated graph is constructed as the union over the set of edges, found by employing the l_1- penalized log-linear regressions.

In detail, let $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(n)}$ be n independent p-random vectors with the node conditional distribution specified in (2.1), where $\mathbf{X}^{(i)} = (X_{i1}, \ldots, X_{ip})$; and $\mathbb{X} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\}$ be the collection of n samples drawn from the random vectors $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(n)}$, with $\mathbf{x}^{(i)} = (x_{i1}, \ldots, x_{ip}), \quad i = 1, \ldots, n$. Let $\mathbb{X}_U$ be the set of n samples of the $|U|$-random vector $\mathbf{X}_U$, with $\mathbf{x}_U^{(i)} = (x_{ij})_{j\in U}, \quad i = 1, \ldots, n$. Since we are only interested in considering the structure, we can drop the parameters $\theta_s, \ s \in V$ for simplicity. Thus, we can rewrite the node conditional distribution as follows

$$X_s|\mathbf{x}_{V\backslash\{s\}} \sim \text{Pois}(\exp\{\sum_{t\neq s}\theta_{st}x_t\}), \quad s \in V = \{1,\dots,p\}. \tag{2.10}$$

Under the assumption (2.10), for each $s \in V$ the node conditional distribution can be written as

$$\begin{aligned}\mathbb{P}_{\boldsymbol{\theta}_{V\backslash\{s\}}}(x_s|\mathbf{x}_{V\backslash\{s\}}) &= \exp\{x_s\sum_{t\neq s}\theta_{st}x_t - \log(x_s!) - e^{\sum_{t\neq s}\theta_{st}x_t}\} \qquad (2.11)\\ &= \exp\{x_s\langle\boldsymbol{\theta}_{V\backslash\{s\}},\mathbf{x}_{V\backslash\{s\}}\rangle + C(x_s) - D(\langle\boldsymbol{\theta}_{V\backslash\{s\}},\mathbf{x}_{V\backslash\{s\}}\rangle)\},\end{aligned}$$

where $\langle .,.\rangle$ denotes the inner product, $\boldsymbol{\theta}_{V\backslash\{s\}} = \{\theta_{st}: \ t\in V\backslash\{s\}\}$, $C(a) = \log(a!), \ a > 0$, and $D(a) = e^a, \ a \in \mathbb{R}$.

Then, a rescaled negative node conditional log-likelihood is as follows

$$\begin{aligned}l(\boldsymbol{\theta}_{V\backslash\{s\}},\mathbb{X}) &= -\frac{1}{n}\ell(\boldsymbol{\theta}_{V\backslash\{s\}},\mathbb{X}) = -\frac{1}{n}\log\prod_{i=1}^{n}\mathbb{P}_{\boldsymbol{\theta}_{V\backslash\{s\}}}(x_{is}|\mathbf{x}^{(i)}_{V\backslash\{s\}}) \qquad (2.12)\\ &= \frac{1}{n}\sum_{i=1}^{n}\left[-x_{is}\langle\boldsymbol{\theta}_{V\backslash\{s\}},\mathbf{x}^{(i)}_{V\backslash\{s\}}\rangle + D(\langle\boldsymbol{\theta}_{V\backslash\{s\}},\mathbf{x}^{(i)}_{V\backslash\{s\}}\rangle)\right],\end{aligned}$$

where $\ell(.)$ is the log-likelihood function. The parameter $\boldsymbol{\theta}_{V\backslash\{s\}}$ is estimated by minimizing the rescaled negative node conditional log-likelihood (2.12), i.e.,

$$\hat{\boldsymbol{\theta}}_{V\backslash\{s\}} = \text{argmin}_{\boldsymbol{\theta}_{V\backslash\{s\}}\in\mathbb{R}^{p-1}} l(\boldsymbol{\theta}_{V\backslash\{s\}},\mathbb{X}). \tag{2.13}$$

To encourage sparsity of estimated graphs, a l_1- regularized conditional log-likelihood is considered, i.e.,

$$\hat{\boldsymbol{\theta}}_{V\backslash\{s\}} = \text{argmin}_{\boldsymbol{\theta}_{V\backslash\{s\}}\in\mathbb{R}^{p-1}} l(\boldsymbol{\theta}_{V\backslash\{s\}},\mathbb{X}) - \lambda\|\boldsymbol{\theta}_{V\backslash\{s\}}\|_1.$$

Given the solution $\hat{\boldsymbol{\theta}}_{V\backslash\{s\}}$, the set of neighbours of node s is defined as

$$\hat{N}(s) = \{t \in V\backslash\{s\}: \quad \hat{\theta}_{st} \neq 0\}.$$

The regularization parameter λ that controls the sparsity of the graph structure is chosen by employing the stability selection criterion (StARS) as in Liu *et al.* (2010), which seeks the regularization parameter λ leading to the most stable set of edges. More precisely, it considers a range $\Lambda = \{\lambda_1,\dots,\lambda_k\}$ of values for λ, and fixes a number m, $1 < m < n$ of observations in one sample. Then, B samples of size m, $S_1,\dots,S_B$, are generated from $\mathbf{x}_1, \ldots, \mathbf{x}_n$. For each $\lambda \in \Lambda$, the graph is estimated by solving a lasso problem. Let $A^m_\lambda(S_1), \ldots, A^m_\lambda(S_B)$ be estimated adjacency matrices of the graph in the subsamples. The stability of one edge can be estimated by

$$\epsilon^m_{s,t}(\lambda) = 2\psi^m_{s,t}(\lambda)\big(1 - \psi^m_{s,t}(\lambda)\big),$$

where $\psi^m_{s,t}(\lambda) = \frac{1}{B}\sum_{i=1}^{B} A^m_\lambda(S_i)_{st}$ is the estimated probability of one edge between nodes s and t. The optimal value λ_{opt} is defined as the largest value that maximizes the total stability

$$\bar{D}_m(\lambda) = sup_{0\leq\rho\leq\lambda} \sum_{s<t} \epsilon^m_{s,t}(\rho) / \binom{p}{2},$$

smaller than a chosen value for the upper bound β, i.e., $\lambda_{opt} = \sup\{\lambda : \bar{D}_B(\lambda) \leq \beta\}$.

2.3.2 PDN Algorithm

The Poisson dependency network (PDN) algorithm, introduced by Hadiji *et al.* (2015), is a nonparametric algorithm for structure learning of both directed and undirected Poisson graphical models. It is a scoring-based algorithm assuming that local conditional distributions follow a Poisson law, but $f_s(.)$ in (2.1) is set to be an arbitrary function. Structure learning is achieved by estimating the mean function in a functional space, employing gradient ascent through gradient tree boosting, and multivariate gradient boosting.

Here, we consider two choices for the link function: log link function and identity function. For more comments on the choice of the link, we refer the interested reader to Hadiji *et al.* (2015).

For the log link function, we can rewrite the mean function as

$$f_s(\mathbf{x}_{V\setminus\{s\}}) = \exp\{\psi_s(\mathbf{x}_{V\setminus\{s\}})\},$$

where $\psi_s(.)$ is some function of $\mathbf{X}_{V\setminus\{s\}}$. In order to estimate $\psi(\mathbf{x}_{V\setminus s})$, a gradient ascent is applied in a function space for some iterations T,

$$f^t_s(\mathbf{x}_{V\setminus\{s\}}) = \exp\{\psi^t_s(\mathbf{x}_{V\setminus\{s\}})\} = \exp\{\psi^{t-1}_s(\mathbf{x}_{V\setminus\{s\}}) + \eta\nabla^t_s\},\ t = 1, \ldots, T,$$

where η, $\eta > 0$ is a step size parameter usually set equal to 1, and the function of gradient is:

$$\nabla_s^t = \frac{1}{n}\sum_{i=1}^{n} \frac{\partial}{\partial \psi_s^{t-1}(\mathbf{x}_{V\setminus\{s\}}^{(i)})} \log \mathbb{P}\left(x_{is}|\mathbf{x}_{V\setminus\{s\}}^{(i)}; \psi_s^{t-1}(\mathbf{x}_{V\setminus\{s\}}^{(i)})\right) = \frac{1}{n}\sum_{i=1}^{n} \nabla_s^t(x_{is}, \mathbf{x}_{V\setminus\{s\}}^{(i)}).$$

The initial $\psi_s^0(\mathbf{x}_{V\setminus s})$ could be fixed to be constant or equal to the logarithm of the empirical mean $\bar{x}_s$ or to a more complex model. This procedure gives a pointwise estimate of $f_s(\mathbf{x}_{V\setminus\{s\}})$. However, the interest is in estimating $f_s(.)$ at arbitrary values $\mathbf{x}$ other than those in the training sample. Hence, by imposing smoothness on the solution, the function estimate can be achieved by borrowing from nearby data points. One way to do it is using regression trees to approximate the true function gradient, i.e., the following objective is minimized

$$\sum_j [h_s^t(\mathbf{x}_{V\setminus\{s\}}^{(i)}) - \nabla_s^t(x_{is}, \mathbf{x}_{V\setminus\{s\}}^{(i)})]^2,$$

where $h_s^t(.)$ is a regression tree function as in Breiman *et al.* (1984). As some choices can lead to really slow convergence, a multiplicative boosting approach with the use of identity link is employed.

For the identity link, i.e., $f_s(.) = \psi_s(.)$, the multiplicative update is specified by the following theorem.

Theorem 2.3.1. The multiplicative functional gradient ascent using the identity link function is

$$\psi^{t+1}(\mathbf{x}_{V\setminus\{s\}}) = \psi^t(\mathbf{x}_{V\setminus\{s\}})\mathbb{E}_{X_s, \mathbf{x}_{V\setminus\{s\}}}\left[\frac{X_s}{f_s(\mathbf{X}_{V\setminus\{s\}})}\right]$$

and the training instance can be generated using $\left(\mathbf{x}_{V\setminus\{s\}}^{(i)}, \frac{x_{is}}{f_s(\mathbf{x}_{V\setminus\{s\}}^{(i)})}\right)$.

The main contribution of this method is the development of the multivariate gradient boosting, which is proved to be efficient in increasing the convergence rate, since it selects the step size automatically without extra computations.

2.3.3 ODS algorithm

The overdispersion score (ODS) algorithm, introduced in Park and Raskutti (2015), is one of the latest algorithms for learning large-scale Poisson DAGs. It belongs to the class of scoring-based algorithms and it works in two steps. Firstly, the topological ordering is estimated by using an overdispersion score function. Once the order is estimated, the problem is reduced to an estimation of p classical penalized regression problems. Details of the ODS algorithm are given in what follows.

Similar to some other structure learning algorithms of DAGs that do not employ prior knowledge on topological ordering, ODS starts from searching the candidate parent sets by employing existing algorithms for learning undirected structures, such as, for example, those in Yang *et al.* (2012), Tsamardinos *et al.* (2006), or Aliferis *et al.* (2003). An estimated parent set is specified by learning a moral graph $G^m = (V, E_u)$ underlying the data. We recall that the moral graph of one DAG is defined as an undirected graph that consists of all edges on the original DAG without direction and edges between variables that share a common child. Then, the algorithm estimates the (causal) ordering of Poisson DAGs using an overdispersion scoring criterion, which is based on this result:

Theorem 2.3.2. Assume that for any $j \in V$, $K \subset pa(j)$ and $S \subset \{1, 2, \ldots, p\} \backslash K$,

$$\mathrm{Var}(f_j(\mathbf{X}_{pa(j)}|\mathbf{X}_S)) > 0.$$

Then Poisson DAG model is identifiable.

Precisely, the ordering begins from the node for which the difference between mean and variance is smallest. To identify the next position in the ordering, the j-th element is chosen from the neighbour set of $(j-1)$-th element, for which the difference between conditional mean and variance on the intersection set between the $(j-1)$ first elements in the order and its neighbours reaches the minimum value. Once the topological ordering is known, the structure learning process boils down to p standard regressions which can be performed by using traditional tools [Friedman *et al.* (2009)].

The score criterion of ODS exploits a simple property of Poisson DAGs, i.e, overdispersion. However, it is also the drawback of this method, since it can not work on data which are not generated from a Poisson distribution.

Chapter 3

Structure Learning of undirected graphs

This chapter intends to develop a framework for structure learning of undirected Poisson graphical models. We propose a new algorithm, called PC-LPGM, designed by exploiting two existing algorithms, i.e., the LPGM algorithm [Allen and Liu (2013)], and the PC algorithm [Spirtes *et al.* (2000)] (see Appendix A for details). Briefly, PC-LPGM employs the local approach of Allen and Liu (2013), i.e., it assumes, as LPGM does, that each node conditional distribution follows a Poisson distribution. Then, the neighbourhood of each node is estimated in turn by a testing procedure on the parameters of the local regressions, following the lines of the PC algorithm.

The remainder of this chapter is as follows. Section 3.1 is devoted to the introduction of the proposed method, Section 3.2 provides a theoretical analysis of the algorithm with detailed proofs. In Section 3.3, we provide some experimental results that illustrate the practical performance of our method. Real data analysis on level III breast cancer miRNA expression is given in Section 3.4. Some discussion is provided in Section 3.5.

3.1 The PC-LPGM algorithm

We are considering the problem of learning an undirected (possibly sparse) graphical structure under model specification (2.10), i.e., each node conditional distribution follows a Poisson distribution,

$$X_s|\mathbf{x}_{V\setminus\{s\}} \sim \text{Pois}\bigg(\exp\Big\{\sum_{t\neq s}\theta_{st}x_t\Big\}\bigg), \quad s \in V = \{1, \dots, p\}. \tag{3.1}$$

Structure learning can be performed by mean of conditional independence tests aimed at identifying the set of non-zero parameters θ_{st}. In the Poisson case, conditional independencies can be inferred from Wald type tests on the parameters θ_{st}. Sample estimates $\hat{\theta}_{st}$ can be obtained within a maximum likelihood approach and test statistics built exploiting the asymptotic normality of the estimators. It has to be said, however, that estimation might be problematic when the number of random variables p is large, possibly larger than n. To face this problem, we borrow the solution offered by the PC algorithm, which relies on controlling the number of variables in the conditional sets, a strategy particularly effective when sparse graphs are under consideration.

Starting from the complete graph, for each s and $t \in V \backslash \{s\}$ and for any set of variables $\mathbf{S} \subset \{1, \ldots, p\} \backslash \{s, t\}$, we test, at some pre-specified significance level, the null hypothesis $H_0 : \theta_{st|\mathbf{K}} = 0$, with $\mathbf{K} = \mathbf{S} \cup \{t\}$. In other words, we test if data support existence of the conditional independence relation $X_s \perp\!\!\!\perp X_t | \mathbf{X_S}$. If the null hypothesis is not rejected, the edge (s, t) is considered to be absent from the graph. A control is operated on the cardinality of the set $\mathbf{S}$ of conditioning variables, which is progressively increased from 0 to $p-2$ or to m, $m < (p-2)$, where m the maximum number of neighbours that one node is allowed to have.

In detail, assume

$$X_s | \mathbf{x_K} \sim \text{Pois}\bigg(\exp\Big\{ \sum_{t \in \mathbf{K}} \theta_{st|\mathbf{K}} x_t \Big\} \bigg), \quad \forall s \in V, \ \mathbf{K} \subset \{1, \ldots, p\} \backslash \{s\}, \tag{3.2}$$

and denote $\boldsymbol{\theta}_{s|\mathbf{K}} = \{\theta_{st|\mathbf{K}} : \quad t \in \mathbf{K}\}$. A rescaled negative node conditional log-likelihood given the conditioning variables $\mathbf{X_K} = \{X_k, \quad k \in \mathbf{K}\}$ can be written as

$$\begin{aligned} l(\boldsymbol{\theta}_{s|\mathbf{K}}, \mathbb{X}_{\{s\}}\,; \mathbb{X}_{\mathbf{K}}) &= -\frac{1}{n} \log \prod_{i=1}^{n} \mathbb{P}_{\boldsymbol{\theta}_{s|\mathbf{K}}}(x_{is} | \mathbf{x}_{\mathbf{K}}^{(i)}) \qquad (3.3) \\ &= \frac{1}{n} \sum_{i=1}^{n} \left[-x_{is} \langle \boldsymbol{\theta}_{s|\mathbf{K}}, \mathbf{x}_{\mathbf{K}}^{(i)} \rangle + D(\langle \boldsymbol{\theta}_{s|\mathbf{K}}, \mathbf{x}_{\mathbf{K}}^{(i)} \rangle) \right], \end{aligned}$$

where the scaling factor is taken for later mathematical convenience. The estimate $\hat{\boldsymbol{\theta}}_{s|\mathbf{K}}$ of the parameter $\boldsymbol{\theta}_{s|\mathbf{K}}$ is determined by minimizing the above given rescaled negative conditional log-likelihood (3.3), i.e.,

$$\hat{\boldsymbol{\theta}}_{s|\mathbf{K}} = \text{argmin}_{\boldsymbol{\theta}_{s|\mathbf{K}} \in \mathbb{R}^{|\mathbf{K}|}} \ l(\boldsymbol{\theta}_{s|\mathbf{K}}, \mathbb{X}_{\{s\}}\,; \mathbb{X}_{\mathbf{K}}). \tag{3.4}$$

A Wald-type test statistic for the hypothesis $H_0 : \theta_{st|\mathbf{K}} = 0$ can be obtained from asymptotic normality of $\hat{\boldsymbol{\theta}}_{s|\mathbf{K}}$, i.e., from the result

$$\sqrt{n}(\hat{\boldsymbol{\theta}}_{s|\mathbf{K}} - \boldsymbol{\theta}_{s|\mathbf{K}}) \xrightarrow{\mathrm{d}} N(\mathbf{0}, I(\boldsymbol{\theta}_{s|\mathbf{K}})^{-1}),$$

where $I(\boldsymbol{\theta}_{s|\mathbf{K}})$ denotes the expected Fisher information matrix, i.e.,

$$I(\boldsymbol{\theta}_{s|\mathbf{K}}) = \mathbb{E}\left[n \frac{\partial^2 l(\boldsymbol{\theta}_{s|\mathbf{K}}, X_s\,; \mathbb{X}_{\mathbf{K}})}{\partial^2 \boldsymbol{\theta}_{s|\mathbf{K}}} \right],$$

which holds under fairly general regularity conditions. The test statistic for the null hypothesis $H_0 : \theta_{st|\mathbf{K}} = 0$ can be obtained on exploiting the marginal asymptotic normality of the component $\hat{\theta}_{st|\mathbf{K}}$.

In practice, the observed information $J(\boldsymbol{\theta}_{s|\mathbf{K}}) = n \dfrac{\partial^2 l(\boldsymbol{\theta}_{s|\mathbf{K}}, \mathbb{X}_{\{s\}}; \mathbb{X}_{\mathbf{K}})}{\partial^2 \boldsymbol{\theta}_{s|\mathbf{K}}}$, i.e., the second derivative of the negative log-likelihood function, is more conveniently used evaluated at $\hat{\boldsymbol{\theta}}_{s|\mathbf{K}}$ as variance estimate of maximum likelihood quantities instead of the expected Fisher information matrix, a modification which comes from the use of an appropriately conditioned sampling distribution for the maximum likelihood estimators. Following this line, the test statistic for the hypothesis $H_0 : \theta_{st|\mathbf{K}} = 0$ is given by

$$Z_{st|\mathbf{K}} = \frac{\sqrt{n}\hat{\theta}_{st|\mathbf{K}}}{\sqrt{\left[J(\hat{\boldsymbol{\theta}}_{s|\mathbf{K}})^{-1} \right]_{tt}}}, \tag{3.5}$$

where $[A]_{jj}$ denotes the element in position (j, j) of matrix A. It is readily available that $Z_{st|\mathbf{K}}$ is asymptotically standard normally distributed under the null hypothesis, provided that some general regularity conditions hold [Lehmann (1986), page 185].

The conditional independence tests are prone to mistakes. Moreover, incorrectly deleting or retaining an edge would result in changes in the neighbour sets of other nodes, as the graph is updated dynamically. Therefore, the resulting graph is dependent on the order in which the conditional independence tests are performed. To avoid this problem, we employ the solution in Colombo and Maathuis (2014), who developed a modification of the PC algorithm that removes the order-dependence, called PC-stable. In this modification, the neighbours of all nodes are searched for and kept unchanged at each particular cardinality l of the set $\mathbf{K}$. As a result, an edge deletion at one level does not affect the conditioning sets of the other nodes, and thus the output is independent on the variable ordering.

The pseudo-code of our algorithm can be written as in Algorithm 1, where $\text{adj}(G, s) = \{t \in G : (s, t) \in E\}$ denotes the set of all nodes that are adjacent to s on the graph G.

Algorithm 1 The PC-LPGM algorithm.

```
 1: Input: $n$ independent realizations of the $p$-random vector $\mathbf{X}$; i.e., $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}$;
    an ordering $order(V)$ on the variables, (and a stopping level $m$).
 2: Output: An estimated undirected graph $\hat{G}$.
 3: Form the complete undirected graph $\tilde{G}$ on the vertex set $V$.
 4: $l = -1$;    $\hat{G} = \tilde{G}$
 5: repeat
 6:    $l = l + 1$
 7:    for all vertices $s \in V$, do
 8:       let $\mathbf{K}_s = \text{adj}(\hat{G}, s)$
 9:    end for
10:    repeat
11:       Select a (new) ordered pair of nodes $s, t$ that are adjacent in $\hat{G}$ s.t.
12:       $|\mathbf{K}_s \backslash \{t\}| \geq l$, using $order(V)$.
13:       repeat
14:          choose a (new) set $\mathbf{S} \subset \mathbf{K}_s \backslash \{t\}$ with $|\mathbf{S}| = l$, using $order(V)$.
15:          if $H_0 : \theta_{st|\mathbf{S}} = 0$ not rejected
16:             delete edge $(s, t)$ from $\hat{G}$
17:          end if
18:       until edge $(s, t)$ is deleted or all $\mathbf{S} \subset \mathbf{K}_s \backslash \{t\}$ with $|\mathbf{S}| = l$ have been considered.
19:    until all ordered pair of adjacent variables $s$ and $t$ such that $|\mathbf{K}_s \backslash \{t\}| \geq l$ and
20:       $\mathbf{S} \subset \mathbf{K}_s \backslash \{t\}$ with $|\mathbf{S}| = l$ have been tested for conditional independence.
21: until $l = m$ or for each ordered pair of adjacent nodes $s, t$: $|\text{adj}(\hat{G}, s) \backslash \{t\}| < l$.
```

We note that the pseudo-code is identical to Algorithm 4.1 in Colombo and Maathuis (2014). Indeed, the difference lies in the statistical procedure used to test the hypothesis at line 15.

3.2 Statistical Guarantees

In this section, we address the property of statistical consistency of our algorithm, i.e., we will study in detail the limiting behavior of our estimation procedure as the sample size n goes to infinity and the model size p remains fixed. It should be noted that we employ maximum likelihood estimation at each local regression, which guarantees consistency and asymptotic normality of the estimators. However, in what follows, we will derive consistency explicitly as a function of the sample size, n, the number of nodes, p, and of the maximum number of neighbours, m. We will begin by stating the assumptions that underlie our analysis, and then give a precise statement of the main result. We acknowledge that our results are based on the work of Yang *et al.* (2015) for exponential family models, combined with ideas coming from Kalisch and Bühlmann (2007). In particular, we adapted to our setting the proof of consistency of estimators in l_1 regularized local models given in Yang *et al.* (2015). Moreover, we exploited the ideas of Kalisch and Bühlmann (2007) for proving consistency of the graph estimator.

For the readers' convenience, before stating the main result, we summarize some notation that will be used through out this proof. Given a vector $v \in \mathbb{R}^p$, and a parameter $q \in [0, \infty]$, we write $\|u\|_q$ to denote the usual l_q norm. Given a matrix $A \in \mathbb{R}^{a \times a}$, denote the largest eigenvalue, and the smallest eigenvalue as $\Lambda_{\max}(A)$, $\Lambda_{\min}(A)$ respectively. We use $|||A|||_2$ to denote the spectral norm, corresponding to the largest singular value of A, i.e.,

$$|||A|||_2 = \sqrt{\Lambda_{\max}(A^T A)},$$

and the l_∞ matrix norm is defined as

$$|||A|||_\infty = \max_{i=1,\dots,a} \sum_{j=1}^{a} |A_{i,j}|.$$

3.2.1 Assumptions

Let $\boldsymbol{\theta}^* = \{\boldsymbol{\theta}^*_{V\backslash\{s\}},\ s \in V\}$ denote the true value of $\boldsymbol{\theta}$. Denote the Hessian matrix of the node rescale negative conditional log-likelihood at $\boldsymbol{\theta}_{V\backslash\{s\}}$ as $Q_s(\boldsymbol{\theta}_{V\backslash\{s\}}) = \nabla^2 l(\boldsymbol{\theta}_{V\backslash\{s\}}, \mathbb{X})$. Note that $Q_s(\boldsymbol{\theta}_{V\backslash\{s\}}) = J(\boldsymbol{\theta}_{V\backslash\{s\}})/n$.

We note that we will consider the problem of maximum likelihood on a closed and bounded dish $\boldsymbol{\Theta} \subset \mathbb{R}^{(p-1)}$. For $\boldsymbol{\theta}_{s|\mathbf{K}} \in \mathbb{R}^{|K|}$, we can immerse $\boldsymbol{\theta}_{s|\mathbf{K}}$ into $\boldsymbol{\Theta} \subset \mathbb{R}^{(p-1)}$ by zero-pad $\boldsymbol{\theta}_{s|\mathbf{K}}$ to include zero weights over $\{V\backslash\mathbf{K}\}$.

We state our assumptions.

Assumption 3.2.1. [Dependency condition] The Hessian matrix corresponding to the covariates in model (3.1) has bounded eigenvalues; that is, there exists a constant $\lambda_{\min} > 0$ s.t.

$$\Lambda_{\min}(Q_s(\boldsymbol{\theta}_{V\backslash\{s\}})) \geq \lambda_{\min},\ \forall\ \boldsymbol{\theta}_{V\backslash\{s\}} \in \boldsymbol{\Theta}.$$

Moreover, we require that

$$\Lambda_{\max}(E_{\boldsymbol{\theta}_{V\backslash\{s\}}}[\mathbf{X}^T_{V\backslash\{s\}}\mathbf{X}_{V\backslash\{s\}}]) \leq \lambda_{\max}, \quad \forall s \in V,\ \forall\ \boldsymbol{\theta}_{V\backslash\{s\}} \in \boldsymbol{\Theta},$$

where $\lambda_{\max}$ is some constant s.t. $\lambda_{\max} < \infty$.

These conditions ensure that the relevant covariates do not become overly dependent.

Assumption 3.2.2. [Sparsity] The coefficients $\boldsymbol{\theta}_{s|\mathbf{K}} \in \boldsymbol{\Theta}$ for all $s \in V$ and all sets $\mathbf{K} \subset \{1, 2, \ldots, p\} \backslash \{s\}$ have an upper bound l_2 norm, i.e.,

$$\sup_{s,\mathbf{K}} \|\boldsymbol{\theta}_{s|\mathbf{K}}\|_2 \leq M < \infty,$$

and a lower bound norm,

$$\inf_{s,t,\mathbf{K}} |\theta_{st|\mathbf{K}}| \geq c, \quad \forall\ \theta_{st|\mathbf{K}} \neq 0,$$

where $t \in \mathbf{K}$.

Assumption 3.2.3. [Regularity] There exist constants κ_1 and κ_2 s.t. $D(\kappa_1 \log \nu) \leq n^{\kappa_2}$, where $\nu = \max\{n, p\}$, $\kappa_1 \geq \frac{9}{2} M$ (where M is specified in Assumption 3.2.2), and $\kappa_2 \in [0, \frac{1}{4}]$.

Assumption 3.2.4. Suppose $\mathbf{X}$ is a p-random vector with node conditional distribution specified in (3.1). Then, for any vector $u \in \mathbb{R}^p$ satisfying $\|u\|_2 \leq a$, and any positive constant δ, there exists some constant $c_1 > 0$, s.t.

$$\mathbb{P}_{\boldsymbol{\theta}}(|\langle u, \mathbf{X} \rangle| \geq \delta \log \nu) \leq c_1 \nu^{-\delta/a},\ \forall\ \boldsymbol{\theta} \in \boldsymbol{\Theta}.$$

Assumption 3.2.5. Suppose $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(n)}$ are n independent p-random vectors with the node conditional distribution specified in (3.1). Then, for any $\delta > 0$, there exists a constant $c_2 > 0$ s.t.

$$\mathbb{P}_{\boldsymbol{\theta}_{V\backslash\{s\}}} \left(\frac{1}{n} \sum_{i=1}^{n} (X_{is})^2 \geq \delta \right) \leq 2\exp(-c_2 n \delta^2), \quad \forall s \in V,\ \forall\ \boldsymbol{\theta}_{V\backslash\{s\}} \in \boldsymbol{\Theta}.$$

It is worth noting that conditions in Assumption 3.2.4, and Assumption 3.2.5 are always satisfied in exponential family models (see Yang *et al.* (2012)). In particular, they hold under the Poisson assumption if and only if $\theta_{st} \leq 0,\ \forall s, t \in V$. As they are the key technical conditions under which we can prove the consistency of our algorithm, we assume them to hold in all other cases. In other words, these conditions specify a class of node conditional Poisson distribution families, for which the method of Structure learning is consistent.

3.2.2 Consistency of estimators in local models

We begin by introducing some results for the case $\mathbf{K} = V\backslash\{s\}$ with precise proofs. Then, the same results for general case $\mathbf{K} \subset V\backslash\{s\}$ are deduced. To maintain the same notation, in what follows, we write $l(\boldsymbol{\theta}_{s|\{V\backslash\{s\}\}}, \mathbb{X}_{\{s\}}; \mathbb{X}_{V\backslash\{s\}})$ as $l(\boldsymbol{\theta}_{V\backslash\{s\}}, \mathbb{X})$.

Proposition 3.2.6. Assume 3.2.2- 3.2.5. Then, for any $\delta > 0$

$$\mathbb{P}_{\boldsymbol{\theta}_{V\setminus\{s\}}}(\|\nabla l(\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbb{X})\|_\infty \geq \delta) \leq \exp(-c_4 n^{1-\kappa_2}) + c_2\nu^{-5/4} + \exp(-c_3 n),\ \forall\ \boldsymbol{\theta}_{V\setminus\{s\}} \in \boldsymbol{\Theta},$$

when $n \longrightarrow \infty$.

Proof. As we have specified the form of the node conditional log-likelihood function $l(\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbb{X})$ in (2.12), then the t-partial derivative of $l(\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbb{X})$ is:

$$\nabla_t l(\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbb{X}) \ = \ \frac{1}{n}\sum_{i=1}^{n}\left[-x_{is}x_{it} + x_{it}D(\langle\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbf{x}^{(i)}_{V\setminus\{s\}}\rangle)\right]$$

Let $V_{is}(t) = X_{is}x_{it} - x_{it}D(\langle\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbf{x}^{(i)}_{V\setminus\{s\}}\rangle)$, then for any constant $h > 0$, we have

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\theta}_{V\setminus\{s\}}}\left[\exp(hV_{is}(t))|\mathbf{x}^{(i)}_{V\setminus\{s\}}\right] &= \sum_{x_{is}=0}^{\infty}\exp\Big\{h[x_{is}x_{it} - x_{it}D(\langle\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbf{x}^{(i)}_{V\setminus\{s\}}\rangle)] \\
&\quad + x_{is}\langle\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbf{x}^{(i)}_{V\setminus\{s\}}\rangle + C(x_{is}) - D(\langle\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbf{x}^{(i)}_{V\setminus\{s\}}\rangle)\Big\} \\
&= \sum_{x_{is}=0}^{\infty}\exp\Big\{x_{is}[hx_{it} + \langle\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbf{x}^{(i)}_{V\setminus\{s\}}\rangle] + C(x_{is}) \\
&\quad - hx_{it}D(\langle\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbf{x}^{(i)}_{V\setminus\{s\}}\rangle) - D(\langle\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbf{x}^{(i)}_{V\setminus\{s\}}\rangle)\Big\} \\
&= \exp\Big\{D(hx_{it} + \langle\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbf{x}^{(i)}_{V\setminus\{s\}}\rangle) - D(\langle\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbf{x}^{(i)}_{V\setminus\{s\}}\rangle) \\
&\quad - hx_{it}D(\langle\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbf{x}^{(i)}_{V\setminus\{s\}}\rangle)\Big\} \\
&= \exp\left\{\frac{h^2}{2}(x_{it})^2 D(vhx_{it} + \langle\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbf{x}^{(i)}_{V\setminus\{s\}}\rangle)\right\},
\end{aligned} \tag{3.6}$$

for some $v \in [0, 1]$, where we move from line 2 to line 3 by applying $e^\lambda = \sum_{x=0}^{\infty}\frac{e^x}{x!}$, and from line 3 to line 4 by using a Taylor expansion for function $D(.)$ at $\langle\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbf{x}^{(i)}_{V\setminus\{s\}}\rangle$.

Define the event $\zeta_1 = \left\{\max_i |vhX_{it} + \langle\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbf{X}^{(i)}_{V\setminus\{s\}}\rangle| \leq \kappa_1 \log\nu\right\}$. We note that $vhX_{it} + \langle\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbf{X}^{(i)}_{V\setminus\{s\}}\rangle$ has the form of $\langle u, \mathbf{X}\rangle$ with $\|u\|_2 \leq 2M$. Let ζ_1^c be the complementary set of ζ_1, then by Assumption 3.2.4

$$\begin{aligned}
\mathbb{P}_{\boldsymbol{\theta}}(\zeta_1^c) &\leq \sum_{i=1}^{n}\mathbb{P}_{\boldsymbol{\theta}}(|vhX_{it} + \langle\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbf{X}^{(i)}_{V\setminus\{s\}}\rangle| > \kappa_1\log\nu) \\
&\leq c_1 n\nu^{-\kappa_1/(2M)} \leq c_1\nu^{-5/4}, \text{ provided that } |h| \leq M,
\end{aligned}$$

since $\kappa_1 \leq \frac{9}{2}M$. Therefore, from (3.6) and Assumption 3.2.3, we obtain

$$\frac{1}{n}\sum_{i=1}^{n} \log \mathbb{E}_{\boldsymbol{\theta}_{V\backslash\{s\}}} \left[\exp(hV_{is}(t))|\mathbf{x}^{(i)}_{V\backslash\{s\}})\right] \leq \frac{n^{\kappa_2}h^2}{2}\frac{1}{n}\sum_{i=1}^{n}(x_{it})^2, \quad \text{for } |h| \leq M, \tag{3.7}$$

with probability at least $1 - c_1\nu^{-5/4}$.

Similarly,

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\theta}_{V\backslash\{s\}}} \left[\exp(-hV_{is}(t))|\mathbf{x}^{(i)}_{V\backslash\{s\}}\right] &= \sum_{x_{is}=0}^{\infty} \exp\Big\{h[-x_{is}x_{it} + x_{it}D(\langle\boldsymbol{\theta}_{V\backslash\{s\}}, \mathbf{x}^{(i)}_{V\backslash\{s\}}\rangle)] \\
&\quad +x_{is}\langle\boldsymbol{\theta}_{V\backslash\{s\}}, \mathbf{x}^{(i)}_{V\backslash\{s\}}\rangle + C(x_{is}) - D(\langle\boldsymbol{\theta}_{V\backslash\{s\}}, \mathbf{x}^{(i)}_{V\backslash\{s\}}\rangle)\Big\} \\
&= \exp\Big\{D(-hx_{it} + \langle\boldsymbol{\theta}_{V\backslash\{s\}}, \mathbf{x}^{(i)}_{V\backslash\{s\}}\rangle) - D(\langle\boldsymbol{\theta}_{V\backslash\{s\}}, \mathbf{x}^{(i)}_{V\backslash\{s\}}\rangle) \\
&\quad +hx_{it}D(\langle\boldsymbol{\theta}_{V\backslash\{s\}}, \mathbf{x}^{(i)}_{V\backslash\{s\}}\rangle\Big\} \\
&= \exp\left\{\frac{h^2}{2}(x_{it})^2 D(-vhx_{it} + \langle\boldsymbol{\theta}_{V\backslash\{s\}}, \mathbf{x}^{(i)}_{V\backslash\{s\}}\rangle)\right\}.
\end{aligned} \tag{3.8}$$

Define the event $\zeta_1' = \Big\{\max_i |-vhX_{it} + \langle\boldsymbol{\theta}_{V\backslash\{s\}}, \mathbf{X}^{(i)}_{V\backslash\{s\}}\rangle| \leq \kappa_1 \log\nu\Big\}$. We also note that $-vhX_{it} + \langle\boldsymbol{\theta}_{V\backslash\{s\}}, \mathbf{X}^{(i)}_{V\backslash\{s\}}\rangle$ has the form of $\langle u, \mathbf{X}\rangle$ with $\|u\|_2 \leq 2M$, then by Proposition 3.2.4

$$\mathbb{P}_{\boldsymbol{\theta}}(\zeta_1'^c) \leq c_1 n\nu^{-\kappa_1/(2\|\boldsymbol{\theta}\|_2)} \leq c_1\nu^{^{-5/4}}, \text{ provided that } |h| \leq M.$$

Therefore, from (3.8) and Assumption 3.2.3, we obtain

$$\frac{1}{n}\sum_{i=1}^{n} \log \mathbb{E}_{\boldsymbol{\theta}_{V\backslash\{s\}}} \left[\exp(-hV_{is}(t)|\mathbf{x}^{(i)}_{V\backslash\{s\}})\right] \leq \frac{n^{\kappa_2}h^2}{2}\frac{1}{n}\sum_{i=1}^{n}(x_{it})^2, \quad \text{for } |h| \leq M, \tag{3.9}$$

with probability at least $1 - c_1\nu^{-5/4}$.

We define the third event $\zeta_2 = \left\{\max_{t\in V\backslash\{s\}} \frac{1}{n}\sum_{i=1}^{n}(X_{it})^2 \leq \kappa_3\right\}$, for some constant κ_3. By using Assumption 3.2.5, we can establish the upper bound of the complementary set of the event ζ_2, i.e. ζ_2^c as union of each t,

$$\begin{aligned}
\mathbb{P}_{\boldsymbol{\theta}}(\zeta_2^c) &\leq \sum_{t\in V\backslash\{s\}} \mathbb{P}_{\boldsymbol{\theta}}\left(\frac{1}{n}\sum_{i=1}^{n}(X_{it})^2 \geq \kappa_3\right) \\
&\leq p\exp\left(-c_2 n\kappa_3^2\right) \\
&= \exp\left(-c_2 n\kappa_3^2 + \log p\right) \leq \exp(-c_3 n),
\end{aligned}$$

as long as $n > \dfrac{2\log p}{c_2\kappa_3}$. Then, by combining with Inequality (3.7), we have

$$\frac{1}{n}\sum_{i=1}^{n}\log \mathbb{E}_{\boldsymbol{\theta}_{V\setminus\{s\}}}\left[\exp(hV_{is}(t))\middle|\mathbf{x}^{(i)}_{V\setminus\{s\}},\zeta_1,\zeta_2\right] \leq \frac{n^{\kappa_2}h^2\kappa_3}{2}, \quad \text{for } |h| \leq M. \tag{3.10}$$

Similarly, we also have

$$\frac{1}{n}\sum_{i=1}^{n}\log \mathbb{E}_{\boldsymbol{\theta}_{V\setminus\{s\}}}\left[\exp(-hV_{is}(t))\middle|\mathbf{x}^{(i)}_{V\setminus\{s\}},\zeta_1',\zeta_2\right] \leq \frac{n^{\kappa_2}h^2\kappa_3}{2}, \quad \text{for } |h| \leq M. \tag{3.11}$$

Applying Chernoff's bound for any $\delta > 0$

$$\begin{aligned}
& \mathbb{P}_{\boldsymbol{\theta}_{V\setminus\{s\}}}\left(\left|\frac{1}{n}\sum_{i=1}^{n}V_{is}(t)\right| > \delta|\zeta_1,\zeta_1',\zeta_2\right) \\
= \ & \mathbb{P}_{\boldsymbol{\theta}_{V\setminus\{s\}}}\left(\frac{1}{n}\sum_{i=1}^{n}V_{is}(t) > \delta|\zeta_1,\zeta_2\right) + \mathbb{P}_{\boldsymbol{\theta}_{V\setminus\{s\}}}\left(-\frac{1}{n}\sum_{i=1}^{n}V_{is}(t) > \delta|\zeta_1',\zeta_2\right) \\
\leq \ & \frac{\mathbb{E}_{\boldsymbol{\theta}_{V\setminus\{s\}}}\left[\prod_{i=1}^{n}\exp\left(hV_{is}(t)\right)|\zeta_1,\zeta_2\right]}{\exp(nh\delta)} + \frac{\mathbb{E}_{\boldsymbol{\theta}_{V\setminus\{s\}}}\left[\prod_{i=1}^{n}\exp\left(-hV_{is}(t)\right)|\zeta_1',\zeta_2\right]}{\exp(nh\delta)} \\
= \ & \frac{\prod_{i=1}^{n}\mathbb{E}_{\boldsymbol{\theta}_{V\setminus\{s\}}}\left[\exp\left(hV_{is}(t)\right)|\zeta_1,\zeta_2\right]}{\exp(nh\delta)} + \frac{\prod_{i=1}^{n}\mathbb{E}_{\boldsymbol{\theta}_{V\setminus\{s\}}}\left[\exp\left(-hV_{is}(t)\right)|\zeta_1',\zeta_2\right]}{\exp(nh\delta)} \\
= \ & \exp\left\{n.\frac{1}{n}\sum_{i=1}^{n}\log \mathbb{E}_{\boldsymbol{\theta}_{V\setminus\{s\}}}\left[\exp\left(hV_{is}(t)\right)|\zeta_1,\zeta_2\right] - nh\delta\right\} \\
& + \exp\left\{n.\frac{1}{n}\sum_{i=1}^{n}\log \mathbb{E}_{\boldsymbol{\theta}_{V\setminus\{s\}}}\left[\exp\left(-hV_{is}(t)\right)|\zeta_1',\zeta_2\right] - nh\delta\right\} \\
\leq \ & 2\exp\left\{n\left(\frac{n^{\kappa_2}h^2\kappa_3}{2} - h\delta\right)\right\}, \quad \text{for } |h| \leq M.
\end{aligned}$$

Let $h = \dfrac{\delta}{n^{\kappa_2}\kappa_3}$, then

$$\mathbb{P}_{\boldsymbol{\theta}_{V\setminus\{s\}}}\left(\left|\frac{1}{n}\sum_{i=1}^{n}V_{is}(t)\right| > \delta|\zeta_1,\zeta_1',\zeta_2\right) \leq 2\exp\left(-\frac{n\delta^2}{2n^{\kappa_2}\kappa_3}\right), \quad \text{for } n \geq \left(\frac{\delta}{\kappa_3 M}\right)^{1/\kappa_2}.$$

A bound for $W = -\nabla l(\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbb{X})$ can be established as follow

$$\begin{aligned}
\mathbb{P}_{\boldsymbol{\theta}_{V\setminus\{s\}}}(\|W\|_\infty > \delta) &= \mathbb{P}_{\boldsymbol{\theta}_{V\setminus\{s\}}}(\max_{t\in V\setminus\{s\}} |\nabla_t l(\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbb{X})| > \delta) \\
&= \mathbb{P}_{\boldsymbol{\theta}_{V\setminus\{s\}}}\left(\max_{t\in V\setminus\{s\}} \left|\frac{1}{n}\sum_{i=1}^{n} V_{is}(t)\right| > \delta\right) \\
&= \mathbb{P}_{\boldsymbol{\theta}_{V\setminus\{s\}}}\left(\max_{t\in V\setminus\{s\}} \left|\frac{1}{n}\sum_{i=1}^{n} V_{is}(t)\right| > \delta|\zeta_1, \zeta_1', \zeta_2\right) + \mathbb{P}_{\boldsymbol{\theta}}(\zeta_1^c) \\
&\quad + \mathbb{P}_{\boldsymbol{\theta}}(\zeta_1'^c) + \mathbb{P}_{\boldsymbol{\theta}}(\zeta_2^c) \\
&\leq 2p\exp\left(-\frac{n\delta^2}{2n^{\kappa_2}\kappa_3}\right) + \mathbb{P}_{\boldsymbol{\theta}}(\zeta_1^c) + \mathbb{P}_{\boldsymbol{\theta}}(\zeta_1'^c) + \mathbb{P}_{\boldsymbol{\theta}}(\zeta_2^c) \\
&\leq \exp(-c_4 n^{1-\kappa_2}) + c_2\nu^{-5/4} + \exp(-c_3 n),
\end{aligned}$$

for $n \geq \max\left\{\left(\frac{\delta}{\kappa_3 M}\right)^{1/\kappa_2}, \left(\frac{4\log(2p)\kappa_3}{\delta^2}\right)^{1/(1-\kappa_2)}\right\}$. □

Theorem 3.2.7. Assume 3.2.1- 3.2.5. Then, there exists a non-negative decreasing sequence $\delta_n \to 0$, s.t.

$$\mathbb{P}_{\boldsymbol{\theta}_{V\setminus\{s\}}}(\|\hat{\boldsymbol{\theta}}_{V\setminus\{s\}} - \boldsymbol{\theta}_{V\setminus\{s\}}\|_2 \leq \delta_n) \geq 1 - \exp(-c_4 n^{1-\kappa_2}) + c_2\nu^{-5/4} + \exp(-c_3 n),\ \forall\, \boldsymbol{\theta}_{V\setminus\{s\}} \in \boldsymbol{\Theta},$$

when $n \longrightarrow \infty$.

Proof. Let $\hat{\mathbf{u}} = \hat{\boldsymbol{\theta}}_{V\setminus\{s\}} - \boldsymbol{\theta}_{V\setminus\{s\}}$, and define $G : \mathbb{R}^{p-1} \longrightarrow \mathbb{R}$ as

$$G(\hat{\mathbf{u}}) = l(\boldsymbol{\theta}_{V\setminus\{s\}} + \hat{\mathbf{u}}, \mathbb{X}) - l(\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbb{X}).$$

In order to show $\|\hat{\mathbf{u}}\|_2 \leq B$, for some radius $B > 0$, it is sufficient to show $G(\mathbf{u}) > 0$, for all $\mathbf{u} \in \mathbb{R}^{p-1}$ s.t. $\|\mathbf{u}\|_2 = B$. Indeed, if $\|\hat{\mathbf{u}}\|_2 > B$ there exists some $t \in [0, 1]$ s.t. $\|t\hat{\mathbf{u}}\|_2 = B$. It is easy to see that $G(\mathbf{0}) = 0$. Moreover, by the definition of $\hat{\boldsymbol{\theta}}_{V\setminus\{s\}}$ in (2.13), $\hat{\mathbf{u}}$ minimizes $G(\mathbf{u})$ so $G(\hat{\mathbf{u}}) \leq 0$. Finally, since $G(\mathbf{u})$ is a convex function, we get

$$G(t\hat{\mathbf{u}} + (1-t)\mathbf{0}) \leq tG(\hat{\mathbf{u}}) + (1-t)G(\mathbf{0}) \leq 0,$$

contradicting to $G(\mathbf{u}) > 0$, for all $\mathbf{u} \in \mathbb{R}^{p-1}$ s.t. $\|\mathbf{u}\|_2 = B$.

Using Taylor expansion of the node conditional log-likelihood at $\boldsymbol{\theta}_{V\setminus\{s\}}$, we have

$$\begin{aligned}
G(\mathbf{u}) &= l(\boldsymbol{\theta}_{V\setminus\{s\}} + \mathbf{u}, \mathbb{X}) - l(\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbb{X}) \qquad (3.12) \\
&= \nabla l(\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbb{X})\mathbf{u}^T + \mathbf{u}[\nabla^2(l(\boldsymbol{\theta}_{V\setminus\{s\}} + v\mathbf{u}, \mathbb{X}))]\mathbf{u}^T,
\end{aligned}$$

for some $v \in [0,1]$. Let

$$\begin{aligned} q &= \Lambda_{\min}(\nabla^2(l(\boldsymbol{\theta}_{V\setminus\{s\}} + v\mathbf{u}, \mathbb{X}))) \\ &\geq \min_{v\in[0,1]} \Lambda_{\min}(\nabla^2(l(\boldsymbol{\theta}_{V\setminus\{s\}} + vu, \mathbb{X}))) \\ &= \min_{v\in[0,1]} \Lambda_{\min}\left[\frac{1}{n}\sum_{i=1}^{n} D(\langle\boldsymbol{\theta}_{V\setminus\{s\}} + v\mathbf{u}, \mathbf{x}^{(i)}_{V\setminus\{s\}}\rangle)(\mathbf{x}^{(i)}_{V\setminus\{s\}})^T\mathbf{x}^{(i)}_{V\setminus\{s\}}\right]. \end{aligned}$$

Using Taylor expansion for $D(\langle\boldsymbol{\theta}_{V\setminus\{s\}} + v\mathbf{u}, \mathbf{x}^{(i)}_{V\setminus\{s\}}\rangle)$ at $\langle\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbf{x}^{(i)}_{V\setminus\{s\}}\rangle$, we have

$$\begin{aligned} &\frac{1}{n}\sum_{i=1}^{n} D(\langle\boldsymbol{\theta}_{V\setminus\{s\}} + v\mathbf{u}, \mathbf{x}^{(i)}_{V\setminus\{s\}}\rangle)(\mathbf{x}^{(i)}_{V\setminus\{s\}})^T\mathbf{x}^{(i)}_{V\setminus\{s\}}) \\ &= \frac{1}{n}\sum_{i=1}^{n} D(\langle\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbf{x}^{(i)}_{V\setminus\{s\}}\rangle)(\mathbf{x}^{(i)}_{V\setminus\{s\}})^T\mathbf{x}^{(i)}_{V\setminus\{s\}} + \\ &\quad \frac{1}{n}\sum_{i=1}^{n} D(\langle\boldsymbol{\theta}_{V\setminus\{s\}} + v'\mathbf{u}, \mathbf{x}^{(i)}_{V\setminus\{s\}}\rangle)[v\mathbf{u}(\mathbf{x}^{(i)}_{V\setminus\{s\}})^T][(\mathbf{x}^{(i)}_{V\setminus\{s\}})^T\mathbf{x}^{(i)}_{V\setminus\{s\}}], \end{aligned}$$

for some $v' \in [0,1]$. Hence,

$$\begin{aligned} q &\geq \Lambda_{\min}\left[\frac{1}{n}\sum_{i=1}^{n} D(\langle\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbf{x}^{(i)}_{V\setminus\{s\}}\rangle)(\mathbf{x}^{(i)}_{V\setminus\{s\}})^T\mathbf{x}^{(i)}_{V\setminus\{s\}}\right] \\ &\quad - \max_{v'\in[0,1]} \left|\left|\left|\frac{1}{n}\sum_{i=1}^{n} D(\langle\boldsymbol{\theta}_{V\setminus\{s\}} + v'\mathbf{u}, \mathbf{x}^{(i)}_{V\setminus\{s\}}\rangle)[\mathbf{u}(\mathbf{x}^{(i)}_{V\setminus\{s\}})^T]\left[(\mathbf{x}^{(i)}_{V\setminus\{s\}})^T\mathbf{x}^{(i)}_{V\setminus\{s\}}\right]\right|\right|\right|_2 \\ &\geq \lambda_{\min} - \max_{v'\in[0,1]} \left|\left|\left|\frac{1}{n}\sum_{i=1}^{n} D(\langle\boldsymbol{\theta}_{V\setminus\{s\}} + v'\mathbf{u}, \mathbf{x}^{(i)}_{V\setminus\{s\}}\rangle)[\mathbf{u}(\mathbf{x}^{(i)}_{V\setminus\{s\}})^T](\mathbf{x}^{(i)}_{V\setminus\{s\}})^T\mathbf{x}^{(i)}_{V\setminus\{s\}}\right|\right|\right|_2. \end{aligned}$$

It remains to control the spectral norm of the matrix

$$A(v') = \frac{1}{n}\sum_{i=1}^{n} D(\langle\boldsymbol{\theta}_{V\setminus\{s\}} + v'\mathbf{u}, \mathbf{x}^{(i)}_{V\setminus\{s\}}\rangle)[\mathbf{u}(\mathbf{x}^{(i)}_{V\setminus\{s\}})^T](\mathbf{x}^{(i)}_{V\setminus\{s\}})^T\mathbf{x}^{(i)}_{V\setminus\{s\}}), \quad \text{for } v' \in [0,1].$$

Define two new events

$$\begin{aligned} \zeta_3 &= \{\max_i |\langle\boldsymbol{\theta}_{V\setminus\{s\}} + v'\mathbf{u}, \mathbf{X}^{(i)}_{V\setminus\{s\}}\rangle| \leq \kappa_1 \log\nu\}, \\ \zeta_4 &= \{\max_{i,s} \|\mathbf{X}^{(i)}_{V\setminus\{s\}}\|_1 \leq 4\log\nu\}. \end{aligned}$$

Similarly to the event ζ_1, we have $\mathbb{P}_{\boldsymbol{\theta}}(\zeta_3^c) \leq c_3\nu^{-5/4}$, provided that $B \leq M$, and, as a consequence, $|D(\langle\boldsymbol{\theta}_{V\setminus\{s\}} + v'\mathbf{u}, \mathbf{x}^{(i)}_{V\setminus\{s\}}\rangle)| \leq n^{\kappa_2}$, with probability at least $1 - c_3^{-5/4}$. Moreover, $\|\mathbf{X}^{(i)}_{V\setminus\{s\}}\|_1$ has the form of $\langle\mathbf{u}, \mathbf{X}\rangle$, so by Assumption 3.2.4, we obtain $\mathbb{P}_{\boldsymbol{\theta}}(\zeta_4^c) \leq$

$c_2 np\nu^{-4} \leq c_2\nu^{-2}$.

Conditioned on ζ_3, ζ_4 we have

$$\begin{aligned} q &\geq \lambda_{\min} - \max_{v' \in [0,1]} \left|\left|\left| \frac{1}{n}\sum_{i=1}^{n} D(\langle \boldsymbol{\theta}_{V\setminus\{s\}} + v'\mathbf{u}, \mathbf{x}^{(i)}_{V\setminus\{s\}}\rangle)[\mathbf{u}(\mathbf{x}^{(i)}_{V\setminus\{s\}})^T](\mathbf{x}^{(i)}_{V\setminus\{s\}})^T \mathbf{x}^{(i)}_{V\setminus\{s\}} \right|\right|\right|_2 \\ &\geq \lambda_{\min} - \max_{v' \in [0,1]} \left| D(\langle \boldsymbol{\theta}_{V\setminus\{s\}} + v'\mathbf{u}, \mathbf{x}^{(i)}_{V\setminus\{s\}}\rangle)\right| \left| \mathbf{u}(\mathbf{x}^{(i)}_{V\setminus\{s\}})^T \right| \left|\left|\left| \frac{1}{n}\sum_{i=1}^{n} (\mathbf{x}^{(i)}_{V\setminus\{s\}})^T \mathbf{x}^{(i)}_{V\setminus\{s\}} \right|\right|\right|_2 \\ &\geq \lambda_{\min} - n^{\kappa_2} 4 \log\nu \sqrt{d} \|\mathbf{u}\|_2 \lambda_{\max} \\ &= \lambda_{\min} - 4\sqrt{d} B \lambda_{\max} n^{\kappa_2} \log\nu \\ &\geq \frac{\lambda_{\min}}{2}, \quad \text{provided that } B < \frac{\lambda_{\min}}{8\sqrt{d}\lambda_{\max} n^{\kappa_2} \log\nu}. \end{aligned}$$

Let $\delta = \frac{\lambda_{\min}}{2} B$. Then, from Proposition 3.2.6, we have

$$\nabla_t l(\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbb{X}) > -\frac{\lambda_{\min}}{2} B,$$

with probability at least $1 - \exp(-c_4 n^{1-\kappa_2}) + c_2\nu^{-5/4} + \exp(-c_3 n)$, provided that

$$n \geq \max\left\{ \left(\frac{\lambda_{\min} B}{2\kappa_3 M}\right)^{1/\kappa_2}, \left(\frac{8\log(2p)\kappa_3}{\lambda^2_{\min} B^2}\right)^{1/(1-\kappa_2)} \right\}.$$

Combining with the inequality of q, we have

$$\begin{aligned} G(\mathbf{u}) &= \nabla l(\boldsymbol{\theta}_{V\setminus\{s\}}, \mathbb{X})\mathbf{u}^T + \mathbf{u}[\nabla^2(l(\boldsymbol{\theta}_{V\setminus\{s\}} + v\mathbf{u}, \mathbb{X}))]\mathbf{u}^T && (3.13) \\ &> \frac{\lambda_{\min}}{2}B^2 + \frac{\lambda_{\min}}{2}B^2 = 0 && (3.14) \end{aligned}$$

provided that $\left(\frac{8\kappa_3\log(2p)}{\lambda^2_{\min} n^{1-\kappa_2}}\right)^{1/2} < B < \frac{\lambda_{\min}}{8\sqrt{d}\lambda_{\max} n^{\kappa_2}\log\nu}$. It means that $\|\hat{\mathbf{u}}\|_2 < B$. When $n \longrightarrow \infty$ we can choose a non- negative decreasing sequence δ_n s.t. $\left(\frac{8\kappa_3\log(2p)}{\lambda^2_{\min} n^{1-\kappa_2}}\right)^{1/2} < \delta_n < \frac{\lambda_{\min}}{8\sqrt{d}\lambda_{max} n^{\kappa_2}\log\nu}$, then

$$\mathbb{P}_{\boldsymbol{\theta}_{V\setminus\{s\}}}(\|\hat{\boldsymbol{\theta}}_{V\setminus\{s\}} - \boldsymbol{\theta}_{V\setminus\{s\}}\|_2 \leq \delta_n) \geq 1 - \exp(-c_4 n^{1-\kappa_2}) + c_2\nu^{-5/4} + \exp(-c_3 n),$$

when $n \longrightarrow \infty$. □

The proof of the following proposition follows the lines of Proposition 3.2.6. We note that the set of explanatory variables $\mathbf{X_K}$ in the generalized linear model X_s given

$\mathbf{X_K}$ does not include variables X_t, with $t \in \{V \backslash \{s\}\} \backslash \mathbf{K}$. Suppose we zero-pad the true parameter $\boldsymbol{\theta}_{s|\mathbf{K}} \in \mathbb{R}^{|\mathbf{K}|}$ to include zero weights over $\{V \backslash \{s\}\} \backslash \mathbf{K}$, then the resulting parameter would lie in $\mathbb{R}^{|p-1|}$. Hence, we have the following proposition.

Proposition 3.2.8. Assume 3.2.2- 3.2.5. Then, for any $\delta > 0$

$$\mathbb{P}_{\boldsymbol{\theta_{s|\mathbf{K}}}}(\|\nabla l(\boldsymbol{\theta}_{s|\mathbf{K}}, \mathbb{X}_{\{s\}}; \mathbb{X}_{\mathbf{K}})\|_\infty \geq \delta) \leq \exp(-c_4 n^{1-\kappa_2}) + c_2 \nu^{-5/4} + \exp(-c_3 n), \ \forall\ \boldsymbol{\theta_{s|\mathbf{K}}} \in \boldsymbol{\Theta},$$

when $n \longrightarrow \infty$.

Theorem 3.2.9. Assume 3.2.1- 3.2.5. Then there exists a non-negative decrease sequence $\delta_n \to 0$, s.t.

$$\mathbb{P}_{\boldsymbol{\theta}_{s|\mathbf{K}}}(\|\hat{\boldsymbol{\theta}}_{s|\mathbf{K}} - \boldsymbol{\theta}_{s|\mathbf{K}}\|_2 \leq \delta_n) \geq 1 - \exp(-c_4 n^{1-\kappa_2}) + c_2 \nu^{-5/4} + \exp(-c_3 n), \ \forall\ \boldsymbol{\theta_{s|\mathbf{K}}} \in \boldsymbol{\Theta},$$

when $n \longrightarrow \infty$.

Proof. Let $\hat{\mathbf{u}} = \hat{\boldsymbol{\theta}}_{s|\mathbf{K}} - \boldsymbol{\theta}_{s|\mathbf{K}}$, and define $G : \mathbb{R}^{|\mathbf{K}|} \longrightarrow \mathbb{R}$ as

$$G(\hat{\mathbf{u}}) = l(\boldsymbol{\theta}_{s|\mathbf{K}} + \hat{\mathbf{u}}, \mathbb{X}_{\{s\}}; \mathbb{X}_{\mathbf{K}}) - l(\boldsymbol{\theta}_{s|\mathbf{K}}, \mathbb{X}_{\{s\}}; \mathbb{X}_{\mathbf{K}}).$$

We take the same way as in Theorem 3.2.7 in order to show $\|\hat{\mathbf{u}}\|_2 \leq B$, for some radius $B > 0$. In detail, we also show $G(\mathbf{u}) > 0$, for all $\mathbf{u} \in \mathbb{R}^{|\mathbf{K}|}$ s.t. $\|\mathbf{u}\|_2 = B$. Recall the conditional log-likelihood function:

$$l(\boldsymbol{\theta}_{s|\mathbf{K}}, \mathbb{X}_{\{s\}}; \mathbb{X}_{\mathbf{K}}) = \frac{1}{n} \sum_{i=1}^{n} \left[-x_{is} \langle \boldsymbol{\theta}_{s|\mathbf{K}}, \mathbf{x}_{\mathbf{K}}^{(i)} \rangle + D(\langle \boldsymbol{\theta}_{s|\mathbf{K}}, \mathbf{x}_{\mathbf{K}}^{(i)} \rangle) \right].$$

Using Taylor expansion of the node conditional log-likelihood at $\boldsymbol{\theta}_{s|\mathbf{K}}$, we have

$$\begin{aligned} G(\mathbf{u}) &= l(\boldsymbol{\theta}_{s|\mathbf{K}} + \mathbf{u}, \mathbb{X}_{\{s\}}; \mathbb{X}_{\mathbf{K}}) - l(\boldsymbol{\theta}_{s|\mathbf{K}}, \mathbb{X}_{\{s\}}; \mathbb{X}_{\mathbf{K}}) \qquad (3.15) \\ &= \nabla l(\boldsymbol{\theta}_{s|\mathbf{K}}, \mathbb{X}_{\{s\}}; \mathbb{X}_{\mathbf{K}}) \mathbf{u}^T + \mathbf{u}[\nabla^2 (l(\boldsymbol{\theta}_{s|\mathbf{K}} + v\mathbf{u}, \mathbb{X}_{\{s\}}; \mathbb{X}_{\mathbf{K}}))]\mathbf{u}^T. \end{aligned}$$

Let

$$\begin{aligned} q &= \Lambda_{\min}(\nabla^2 (l(\boldsymbol{\theta}_{s|\mathbf{K}} + v\mathbf{u}, \mathbb{X}_{\{s\}}; \mathbb{X}_{\mathbf{K}}))) \\ &\geq \min_{v \in [0,1]} \Lambda_{\min}(\nabla^2 (l(\boldsymbol{\theta}_{s|\mathbf{K}} + vu, \mathbb{X}_{\{s\}}; \mathbb{X}_{\mathbf{K}}))) \\ &= \min_{v \in [0,1]} \Lambda_{\min} \left[\frac{1}{n} \sum_{i=1}^{n} D(\langle \boldsymbol{\theta}_{s|\mathbf{K}} + v\mathbf{u}, \mathbf{x}_{s|\mathbf{K}}^{(i)} \rangle)(\mathbf{x}_{s|\mathbf{K}}^{(i)})^T \mathbf{x}_{s|\mathbf{K}}^{(i)} \right]. \end{aligned}$$

Using Taylor expansion for $D(\langle \boldsymbol{\theta}_{s|\mathbf{K}} + v\mathbf{u}, \mathbf{x}^{(i)}_{s|\mathbf{K}}\rangle)$ at $\langle \boldsymbol{\theta}_{s|\mathbf{K}}, \mathbf{x}^{(i)}_{s|\mathbf{K}}\rangle$, we have

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n} D(\langle \boldsymbol{\theta}_{s|\mathbf{K}} + v\mathbf{u}, \mathbf{x}^{(i)}_{s|\mathbf{K}}\rangle)(\mathbf{x}^{(i)}_{s|\mathbf{K}})^T \mathbf{x}^{(i)}_{s|\mathbf{k}}) &= \frac{1}{n}\sum_{i=1}^{n} D(\langle \boldsymbol{\theta}_{s|\mathbf{K}}, \mathbf{x}^{(i)}_{s|\mathbf{K}}\rangle)(\mathbf{x}^{(i)}_{s|\mathbf{K}})^T \mathbf{x}^{(i)}_{s|\mathbf{K}} \\ &\quad + \frac{1}{n}\sum_{i=1}^{n} D(\langle \boldsymbol{\theta}_{s|\mathbf{K}} + v'\mathbf{u}, \mathbf{x}^{(i)}_{s|\mathbf{K}}\rangle)[v\mathbf{u}(\mathbf{x}^{(i)}_{s|\mathbf{K}})^T] \\ &\qquad [(\mathbf{x}^{(i)}_{s|\mathbf{K}})^T \mathbf{x}^{(i)}_{s|\mathbf{K}}].\end{aligned}$$

Hence,

$$\begin{aligned} q &\geq \Lambda_{\min}\left[\frac{1}{n}\sum_{i=1}^{n} D(\langle \boldsymbol{\theta}_{s|\mathbf{K}}, \mathbf{x}^{(i)}_{s|\mathbf{K}}\rangle)(\mathbf{x}^{(i)}_{s|\mathbf{K}})^T \mathbf{x}^{(i)}_{s|\mathbf{Kd}}\right] \\ &\quad - \max_{v'\in[0,1]} \left|\left|\left|\frac{1}{n}\sum_{i=1}^{n} D(\langle \boldsymbol{\theta}_{s|\mathbf{K}} + v'\mathbf{u}, \mathbf{x}^{(i)}_{s|\mathbf{K}}\rangle)[\mathbf{u}(\mathbf{x}^{(i)}_{s|\mathbf{K}})^T]\,[(\mathbf{x}^{(i)}_{s|\mathbf{K}})^T \mathbf{x}^{(i)}_{s|\mathbf{K}}]\right|\right|\right|_2 \\ &\geq \lambda_{\min} - \max_{v'\in[0,1]} \left|\left|\left|\frac{1}{n}\sum_{i=1}^{n} D(\langle \boldsymbol{\theta}_{s|\mathbf{K}} + v'\mathbf{u}, \mathbf{x}^{(i)}_{s|\mathbf{K}}\rangle)[\mathbf{u}(\mathbf{x}^{(i)}_{s|\mathbf{K}})^T](\mathbf{x}^{(i)}_{s|\mathbf{K}})^T \mathbf{x}^{(i)}_{s|\mathbf{K}}\right|\right|\right|_2 .\end{aligned}$$

The second inequality is due to the fact that the smallest eigenvalue of a sub-matrix is larger than or equal to the smallest eigenvalue of the original matrix. Here, $Q_{s|\mathbf{K}}(\boldsymbol{\theta}_{s|\mathbf{K}}) = \frac{1}{n}\sum_{i=1}^{n} D(\langle \boldsymbol{\theta}_{s|\mathbf{K}}, \mathbf{x}^{(i)}_{V\setminus s}\rangle)(\mathbf{x}^{(i)}_{s|\mathbf{K}})^T \mathbf{x}^{(i)}_{s|\mathbf{K}}$ is a sub-matrix of the Hessian matrix $Q_s(\boldsymbol{\theta}_{V\setminus\{s\}})$. Hence,

$$\Lambda_{\min}\left[\frac{1}{n}\sum_{i=1}^{n} D(\langle \boldsymbol{\theta}_{s|\mathbf{K}}, \mathbf{x}^{(i)}_{s|\mathbf{K}}\rangle)(\mathbf{x}^{(i)}_{s|\mathbf{K}})^T \mathbf{x}^{(i)}_{s|\mathbf{K}}\right] \geq \Lambda_{\min}(Q_s(\boldsymbol{\theta}_{V\setminus\{s\}})) \geq \lambda_{\min}.$$

Then, by performing the same analysis as in the proof of Theorem 3.2.7 and Proposition 3.2.8, we get the result.

□

3.2.3 Consistency of the graph estimator

Now we state the main result of this work for the consistency of the graph estimate.

Assumption 3.2.10. [Faithfulness] Let $J(G)$ be the independence model induced by the true graph G, and J be the independence model induced by conditional independence relations on p random variables $X_1, \ldots, X_p$ in the model (3.1). It holds $J = J(G)$.

Theorem 3.2.11. Assume 3.2.1- 3.2.5, and 3.2.10. Denote by $\hat{G}(\alpha_n)$ the estimate from Algorithm 1, and by G the true graph. Then, there exists a numerical sequence

$\alpha_n \longrightarrow 0$, s.t.

$$\mathbb{P}_{\boldsymbol{\theta}}(\hat{G}(\alpha_n) = G) = 1, \ \forall\ \boldsymbol{\theta} \in \boldsymbol{\Theta},$$

when $n \longrightarrow \infty$.

Proof. Let $\hat{\theta}_{st|\mathbf{K}}$, and $\theta^*_{st|\mathbf{K}}$ denote the estimated and true partial weights between X_s and X_t given $X_r, r \in \mathbf{S}$, where $\mathbf{S} = \mathbf{K}\backslash\{t\} \subset \{1, \ldots, p\}\backslash\{s, t\}$. Many partial weights are tested for being zero during the run of the PC-procedure. For a fixed ordered pair of nodes s, t, the conditioning sets are elements of

$$K^m_{st} = \{\mathbf{S} \subset \{1, \ldots, p\}\backslash\{s, t\} : |\mathbf{S}| \leq m\}.$$

The cardinality is bounded by

$$|K^m_{st}| \leq Bp^m, \qquad \text{for some } 0 < B < \infty.$$

Let $E_{st|\mathbf{K}}$ denote type I or type II errors occurring when testing $H_0 : \ \theta_{st|\mathbf{K}} = 0$. Thus

$$E_{st|\mathbf{K}} = E^I_{st|\mathbf{K}} \cup E^{II}_{st|\mathbf{K}}, \tag{3.16}$$

in which, for n large enough

- type I error $E^I_{st|\mathbf{K}}$: $Z_{st|\mathbf{K}} > \Phi^{-1}(1 - \alpha/2)$ and $\theta^*_{st|\mathbf{K}} = 0$;
- type II error $E^{II}_{st|\mathbf{K}}$: $Z_{st|\mathbf{K}} \leq \Phi^{-1}(1 - \alpha/2)$ and $\theta^*_{st|\mathbf{K}} \neq 0$;

where $Z_{st|\mathbf{K}}$ was defined in (3.5), and α is a chosen significance level. Consider an arbitrary value $\boldsymbol{\theta}_{s|\mathbf{K}} = \{\theta_{sk|\mathbf{K}}\}_{k\in\mathbf{K}} \in \boldsymbol{\Theta}$, and let $\boldsymbol{\theta}^0_{s|\mathbf{K}}$ be the vector that has the same elements as $\boldsymbol{\theta}_{s|\mathbf{K}}$ except $\theta_{st|\mathbf{K}} = \theta^*_{st|\mathbf{K}} = 0$. Choose $\alpha_n = 2(1 - \Phi(n^d))$, with $0 < d < 1/2$, then

$$\begin{aligned}
\sup_{s,t,\mathbf{K}\in K^m_{ij}} \mathbb{P}_{\boldsymbol{\theta}^0_{s|\mathbf{K}}}(E^I_{st|\mathbf{K}}) &= \sup_{s,t,\mathbf{K}\in K^m_{st}} \mathbb{P}_{\boldsymbol{\theta}^0_{s|\mathbf{K}}}\left(|\hat{\theta}_{st|\mathbf{K}}| > n^{d-1/2}\sqrt{\left[J(\hat{\boldsymbol{\theta}}_{s|\mathbf{K}})^{-1}\right]_{tt}}\right) \\
&= \sup_{s,t,\mathbf{K}\in K^m_{st}} \mathbb{P}_{\boldsymbol{\theta}^0_{s|\mathbf{K}}}\left(|\hat{\theta}_{st|\mathbf{K}} - \theta^0_{st|\mathbf{K}}| > n^{d-1/2}\sqrt{\left[J(\hat{\boldsymbol{\theta}}_{s|\mathbf{K}})^{-1}\right]_{tt}}\right) \\
&\leq \exp(-c_4 n^{1-\kappa_2}) + c_2\nu^{-5/4} + \exp(-c_3 n),
\end{aligned} \tag{3.17}$$

using Theorem 3.2.9 and the fact that $n^{d-1/2}\sqrt{\left[J(\hat{\boldsymbol{\theta}}_{s|\mathbf{K}})^{-1}\right]_{tt}} \longrightarrow 0$ as $n \longrightarrow \infty$. Furthermore, with the choice of α_n above,

$$\begin{aligned}
\sup_{s,t,\mathbf{K}\in K_{st}^m} \mathbb{P}_{\boldsymbol{\theta}_{s|\mathbf{K}}}(E_{st|\mathbf{K}}^{II}) &= \sup_{s,t,\mathbf{K}\in K_{st}^m} \mathbb{P}_{\boldsymbol{\theta}_{s|\mathbf{K}}}\left(|\hat{\theta}_{ij|\mathbf{K}}| \le n^{d-1/2}\sqrt{\left[J(\hat{\boldsymbol{\theta}}_{s|\mathbf{K}})^{-1}\right]_{tt}}\right) \\
&< \sup_{s,t,\mathbf{K}\in K_{st}^m} \mathbb{P}_{\boldsymbol{\theta}_{s|\mathbf{K}}}\left(|\hat{\theta}_{st|\mathbf{K}} - \theta_{st|\mathbf{K}}| \ge n^{d-1/2}\sqrt{\left[J(\hat{\boldsymbol{\theta}}_{s|\mathbf{K}})^{-1}\right]_{tt}}\right),
\end{aligned}$$

because $\inf_{s,t,\mathbf{K}} |\theta_{st|\mathbf{K}}| \ge c \ge 2n^{d-1/2}\sqrt{\left[J(\hat{\boldsymbol{\theta}}_{s|\mathbf{K}})^{-1}\right]_{tt}}$, Hence,

$$\begin{aligned}
|\hat{\theta}_{st|\mathbf{K}} - \theta_{st|\mathbf{K}}| &\ge |\theta_{st|\mathbf{K}}| - |\hat{\theta}_{st|\mathbf{K}}| \\
&\ge 2n^{d-1/2}\sqrt{\left[J(\hat{\boldsymbol{\theta}}_{s|\mathbf{K}})^{-1}\right]_{tt}} - n^{d-1/2}\sqrt{\left[J(\hat{\boldsymbol{\theta}}_{s|\mathbf{K}})^{-1}\right]_{tt}} \\
&= n^{d-1/2}\sqrt{\left[J(\hat{\boldsymbol{\theta}}_{s|\mathbf{K}})^{-1}\right]_{tt}}
\end{aligned}$$

Finally, by Theorem 3.2.9, we then obtain

$$\sup_{s,t,\mathbf{K}\in K_{st}^m} \mathbb{P}_{\boldsymbol{\theta}_{s|\mathbf{K}}}(E_{st|\mathbf{K}}^{II}) \le \exp(-c_4 n^{1-\kappa_2}) + c_2\nu^{-5/4} + \exp(-c_3 n), \tag{3.18}$$

as $n \longrightarrow \infty$.

Now, by (3.16)-(3.18), we get

$$\begin{aligned}
&\mathbb{P}_{\boldsymbol{\theta}}(\text{ a type I or II error occurs in testing procedure}) \qquad (3.19)\\
&\le \mathbb{P}_{\boldsymbol{\theta}_{s|\mathbf{K}}}(\cup_{s,t,\mathbf{K}\in K_{st}^m} E_{st|\mathbf{K}}) \qquad (3.20)\\
&\le O(p^{m+2}) \sup_{s,t,\mathbf{K}\in K_{st}^m} \mathbb{P}_{\boldsymbol{\theta}_{s|\mathbf{K}}}(E_{st|\mathbf{K}}) \\
&\le O(p^{m+2})\big[\exp(-c_4 n^{1-\kappa_2}) + c_2\nu^{-5/4} + \exp(-c_3 n)\big] \\
&\longrightarrow 0.
\end{aligned}$$

as $n \longrightarrow \infty$. □

3.3 Empirical study

In this section, we study the performances of our proposed algorithm by means of simulated data.

Simulation studies aim at measuring the ability of PC-LPGM to recover the true structure of the graphs, also in situations where relatively moderate sample sizes are available. As measure of ability, we adopt the positive predictive value and the sensitivity of the algorithm. Let $\hat{G} = (V, \hat{E})$ be the estimate of $G = (V, E)$. The number of true positive (TP), false positive (FP), and false negative (FN) edges are defined to be

$$TP = \#\{\text{ number of edges in } \hat{E} \text{ in } E\},$$

$$FP = \#\{\text{ number of edges in } \hat{E} \text{ not in } E\},$$

$$FN = \#\{\text{ number of edges in } E \text{ not in } \hat{E}\},$$

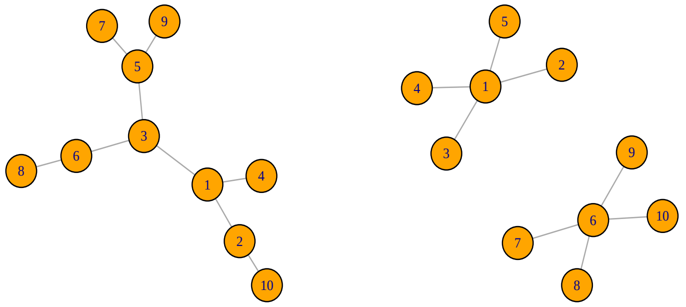
respectively. Then, the positive predictive value (PPV) and the sensitivity (Se) are respectively defined as

$$PPV = \frac{TP}{(TP + FP)}, \quad Se = \frac{TP}{(TP + FN)}.$$

In doing these studies, we also aim to compare PC-LPGM to a number of popular structure learning algorithms. We therefore consider LPGM and PDN. It is worth remembering that structure learning for Poisson undirected graphical models is usually performed by employing methods for continuous data after proper data transformation. We therefore consider two representatives of approaches based on the Gaussian assumption, i.e., variable selection with lasso (VSL) [Meinshausen and Bühlmann (2006)], and graphical lasso algorithm (GLASSO) [Friedman *et al.* (2008)]. Moreover, we consider two structure learning methods dealing with the class of nonparanormal distributions, i.e., the nonparanormal-Copula algorithm (NPN-Copula) [Liu *et al.* (2009)], and the nonparanormal-SKEPTIC algorithm (NPN-Skeptic) [Liu *et al.* (2012)]. Details of considered algorithms are briefly given in Appendix A.

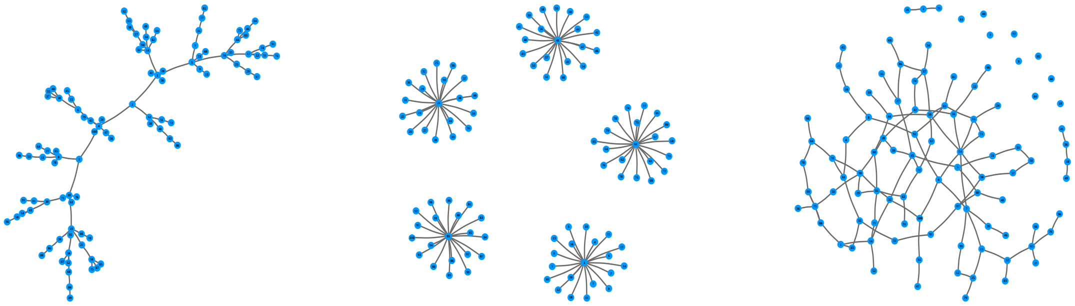
Data generation

For two different cardinalities, i.e., $p = 10$ and $p = 100$, we consider three graphs of different structure: (i) a scale-free graph, in which the node degree distribution follows a powerlaw; (ii) a hub graph, where each node is connected to one of the hub nodes; (iii) a random graph, where presence of edges are i.i.d. Bernoulli random variables. To construct the scale-free and hub networks, we employed the `R` package `XMRF`. For the scale-free network, we assumed a power law with parameter 0.01 for the node degree distribution. For the hub network, we assumed two hub nodes for $p = 10$, and 5 hub nodes for $p = 100$. To construct the random network, we employed the `R` package `igraph` with edge probability 0.2 for $p = 10$, and 0.02 for $p = 100$. See Figure 3.1 and 3.2 for a plot of the three chosen graphs for $p = 10$ and $p = 100$, respectively.



(a) scale−free (b) hub (a) random

FIGURE 3.1: The graph structures for $p = 10$ employed in the simulation studies: (a) scale-free; (b) hub; (c) random graph.



(a) scale−free (b) hub (c) random

FIGURE 3.2: The graph structures for $p = 100$ employed in the simulation studies: (a) scale-free; (b) hub; (c) random graph.

For each graph, 500 datasets were sampled for three sample sizes, i.e., $n = 200, 1000, 2000$. To generate the data, we followed the approach in Allen and Liu (2013). Let $\mathbb{X} = (\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)})^T \in \mathbb{R}^{n \times p}$ be the set of n independent observations of random vector $\mathbf{X}$, where $\mathbf{x}^{(i)}$ is a p-dimensional count data vector, $\mathbf{x}^{(i)} \in \{0, 1, \dots, \infty\}^p$. Then, $\mathbb{X}$ is obtained from the following model

$$\mathbb{X} = \mathbb{Y}W + \epsilon,$$

where $\mathbb{Y} = (y_{st})$ is an $n \times (p + p(p-1)/2)$ matrix whose entries y_{st} are realizations of independent random variables $Y_{st} \sim \text{Pois}(\lambda_{true})$ and $\epsilon = (e_{st})$ is an $n \times p$ matrix with entries e_{st} which are realizations of random variables $E_{st} \sim \text{Pois}(\lambda_{noise})$. Let W be the adjacency matrix of a given true graph, then the adjacency matrix is encoded by matrix W as $W = [I_p; P \odot (1_p tri(W)^T)]^T$. Here, P is a $p \times (p(p-1)/2)$ pairwise permutation matrix, $\odot$ denotes the element-wise product, and $tri(W)$ is the $(p(p-1)/2) \times 1$ vectorized upper triangular part of W. As in Allen and Liu (2013), we simulated data at two signal-to-noise ratio (SNR) levels. We set $\lambda_{true} = 1$ with $\lambda_{noise} = 5$ for the low SNR level, and $\lambda_{noise} = 0.5$ for the high SNR level.

Results

The considered algorithms are listed below, along with specifications, if needed, of tuning parameters. Algorithms for Gaussian data have been used on log transformed data shifted by 1. Whenever a regularization parameter λ had to be chosen, the StARS algorithm Liu *et al.* (2010) was employed, which aims to seek the value of $\lambda \in (\lambda_{min}, \lambda_{max})$, λ_{opt} say, leading to the most stable set of edges. We refer the reader to Section 2.3.1 for details on the StARS algorithm and its tuning parameters, in particular the variability threshold β and the number of subsamplings B. It is worth noting that, whenever the graph corresponding to λ_{opt} was empty, we shifted to the fist nonempty graph (if it existed) in the decreasing regularization path. We therefore considered:

- **PC-LPGM:** level of significance of tests 1%;
- **LPGM:** $\beta = 0.05$; $B = 20$; $\dfrac{\lambda_{min}}{\lambda_{max}} = 0.01$; $\gamma = 0.001$;
- **VSL:** $\beta = 0.1$; $B = 20$;
- **GLASSO:** $\beta = 0.1$; $B = 20$;
- **NPN-Copula:** $\beta = 0.1$; $B = 20$;
- **NPN-skeptic:** $\beta = 0.1$; $B = 20$.

For the two considered vertex cardinalities, i.e., $p = 10, 100$, and for the chosen sample sizes $n = 200, 1000, 2000$, Table 3.1 and Table 3.2 report, respectively, Monte Carlo means of TP, FP, FN, PPV and Se for each of considered method at low ($\lambda_{noise} = 5$) and high ($\lambda_{noise} = 0.5$) SNR levels. Each value is computed as an average of the 1500 values obtained by simulating 500 samples for each of the three networks. Results disaggregated by network type are given in Appendix B, Tables B.1 – B.4. These results indicate that the PC-LPGM algorithm outperforms, on average, Gaussian-based competitors (VSL, GLASSO), nonparanormal-based competitors (NPN-Copula, NPN-Skeptic) as well as the state-of-the-art algorithms that are designed specifically for Poisson graphical models (LPGM, PDN) on average in terms of reconstructing the structure from given data.

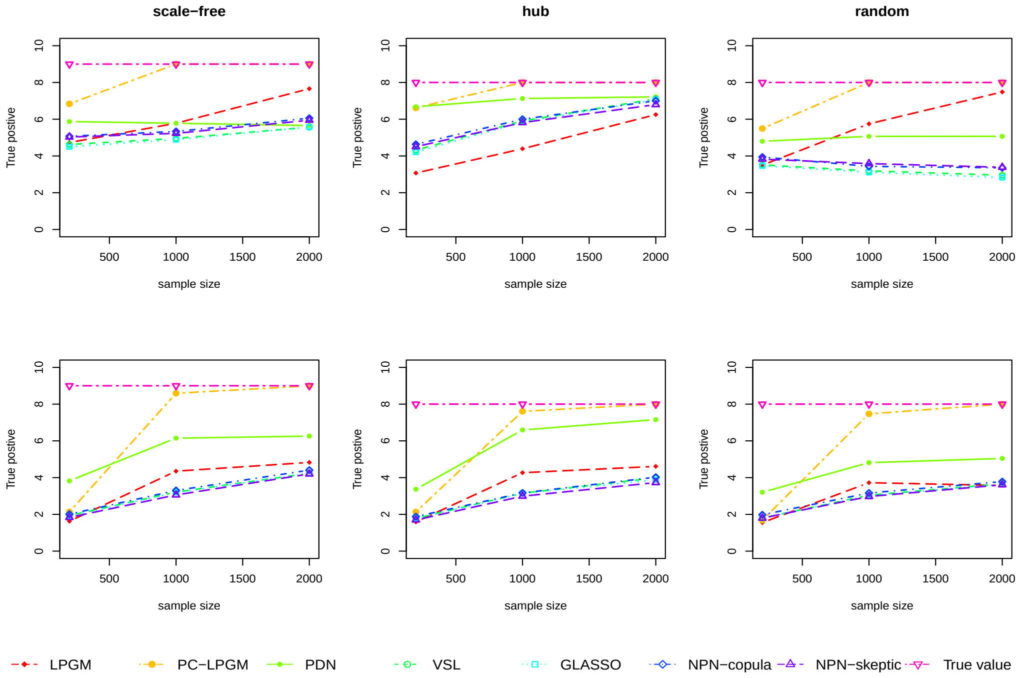


FIGURE 3.3: Number of TP edges recovered by PC-LPGM; LPGM; PDN; VSL; GLASSO; NPN-Copula; NPN-Skeptic for networks in Figure 3.1 ($p = 10$) and sample sizes n = 200, 1000, 2000. First panel row corresponds to high SNR level (λ_{noise} = 0.5); second panel row corresponds to low SNR level ($\lambda_{noise} = 5$).

When $p = 10$, the PC-LPGM algorithm reaches the highest TP value, followed by the PDN and the LPGM algorithms. When $n \geq 1000$, PC-LPGM recovers almost all edges for both low and high SNR levels, see Figure 3.3. A closer look at the PPV and Se plot (see Figure 3.4 and Figure 3.5) provides further insight of the behaviour of considered methods. Among the algorithms with highest PPV, PC-LPGM shows a sensitivity approaching 1 already at the sample size $n = 1000$ for both a high and a low SNR level (Figure 3.4). It is worth noting that LPGM algorithm was successful only for a high SNR level ($\lambda_{nois} = 0.5$).

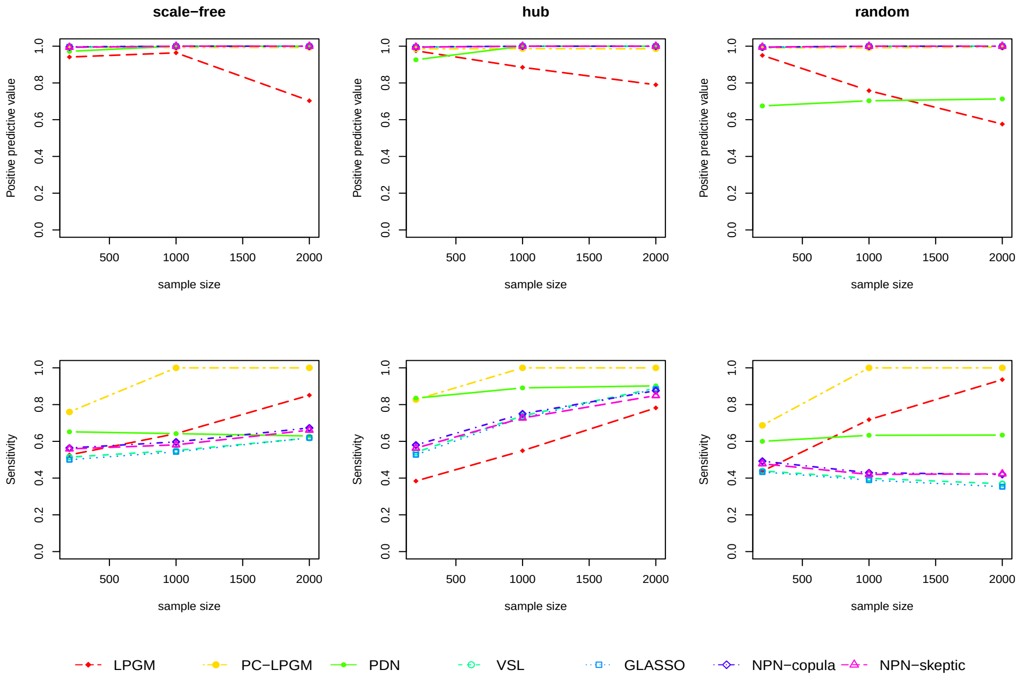


FIGURE 3.4: PPV (first panel row) and Se (second panel row) for PC-LPGM; LPGM; PDN; VSL; GLASSO; NPN-Copula; NPN-Skeptic for networks in Figure 3.1 ($p = 10$), sample sizes $n = 200$, 1000, 2000 and $\lambda_{noise} = 0.5$.

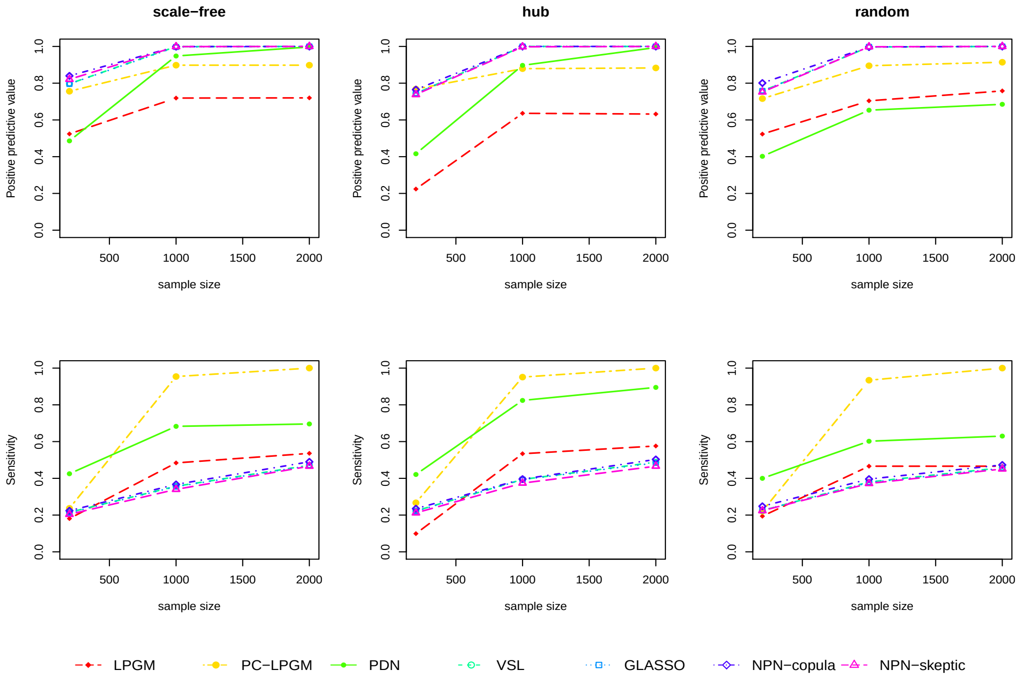


FIGURE 3.5: PPV (first panel row) and Se (second panel row) for PC-LPGM; LPGM; PDN; VSL; GLASSO; NPN-Copula; NPN-Skeptic for networks in Figure 3.1 ($p = 10$), sample sizes $n = 200$, 1000, 2000 and $\lambda_{noise} = 5$.

It is interesting to note that the performance of the PC-LPGM algorithm is far better than that of the competing algorithms employing the Poisson assumption, i.e., PDN and LPGM. This might be explained in terms of difference between penalization and restriction of the conditional sets. In the LPGM algorithm, as well as in the PDN algorithm, a prediction model is fitted locally on all other variables, by mean of a series of independent penalized regressions. In the PC-LPGM algorithm, the number of variables in the conditional sets is controlled and progressively increased from 0 to $p-2$ (or to the maximum number of neighbors m). In our simulations, this second strategy appears to be more powerful in the network reconstruction.

The Gaussian based methods (VSL, GLASSO) perform reasonably well, with an inferior score with respect to the leading threesome only for the hub graph at high SNR level. It is worth noting that sophisticated techniques that replace the Gaussian distribution with a more flexible continuous distribution such as the nonparanormal distribution, e.g., NPN-Copula, NPN-skeptic show slight gains in accuracy over the naive analysis.

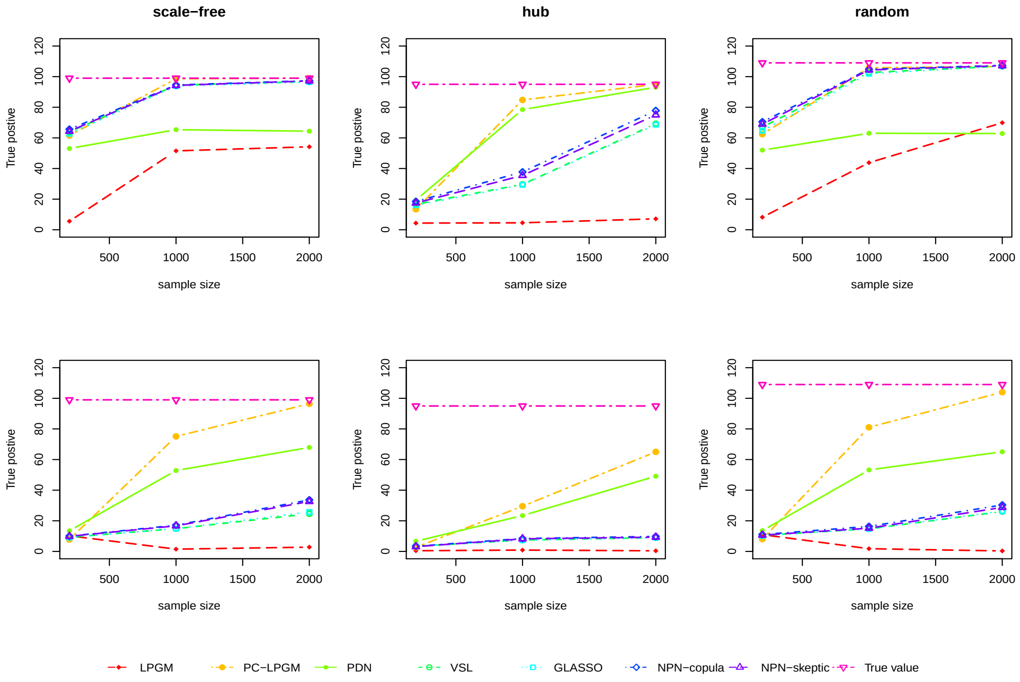


FIGURE 3.6: Number of TP edges recovered by PC-LPGM; LPGM; PDN; VSL; GLASSO; NPN-Copula; NPN-Skeptic for networks in Figure 3.2 ($p = 100$) and sample sizes $n = 200, 1000, 2000$. First panel row corresponds to high SNR level ($\lambda_{noise} = 0.5$); second panel row corresponds to low SNR level ($\lambda_{noise} = 5$).

Results for the high dimensional setting ($p = 100$) are somehow comparable, as it can be seen in Figures 3.6, 3.7 and 3.8. The PC-LPGM outperforms all competing

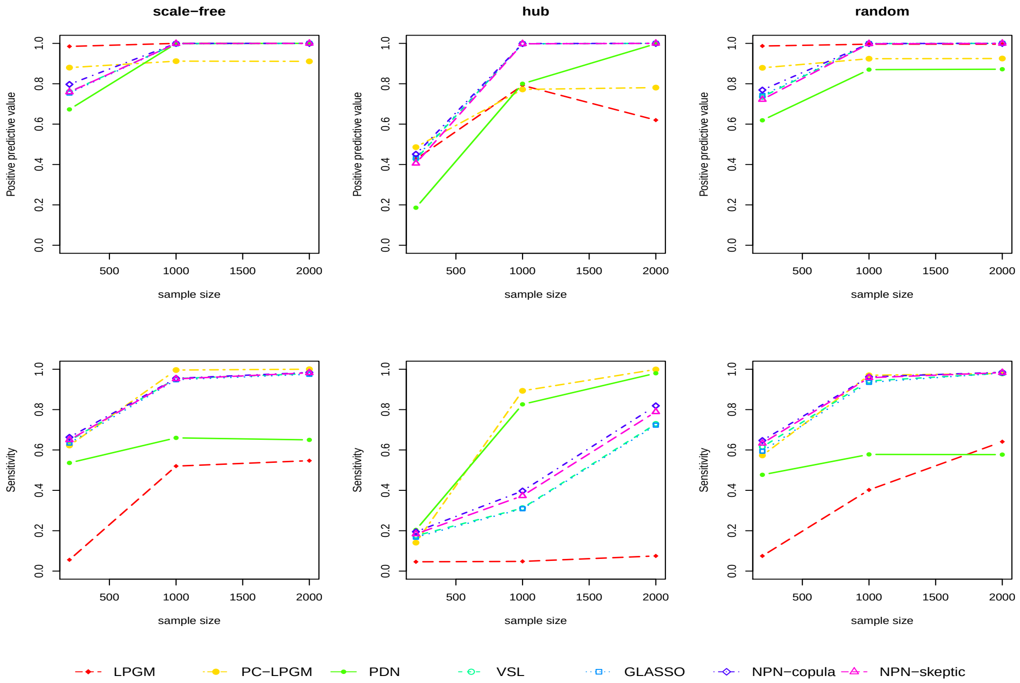


FIGURE 3.7: PPV (first panel row) and Se (second panel row) for PC-LPGM; LPGM; PDN; VSL; GLASSO; NPN-Copula; NPN-Skeptic for networks in Figure 3.2 ($p = 100$), sample sizes $n = 200$, 1000, 2000 and $\lambda_{noise} = 0.5$.

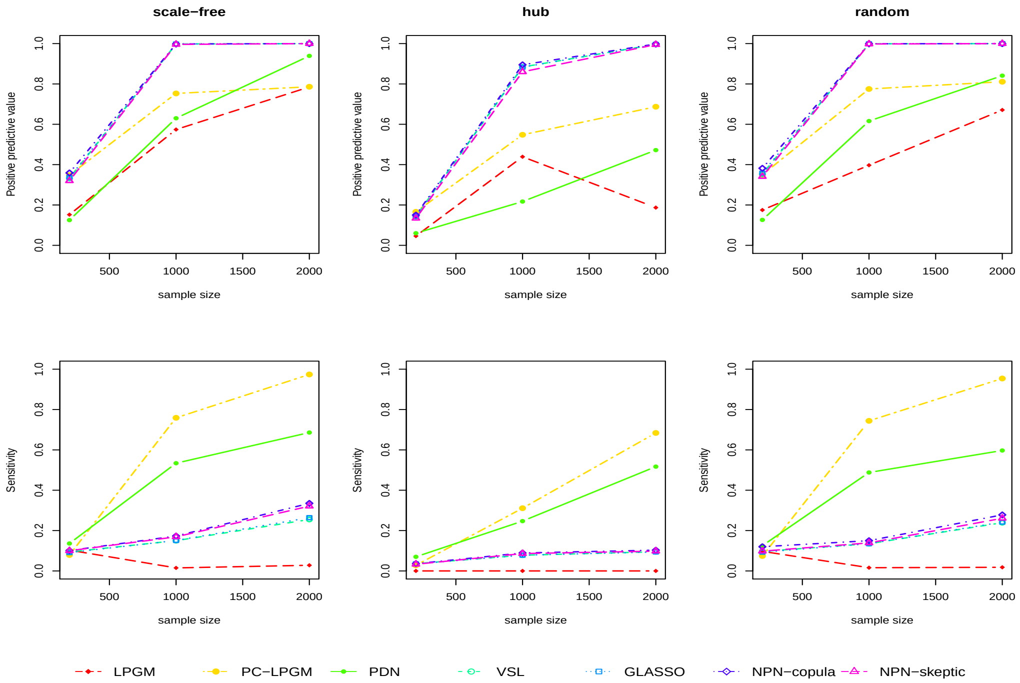


FIGURE 3.8: PPV (first panel row) and Se (second panel row) for PC-LPGM; LPGM; PDN; VSL; GLASSO; NPN-Copula; NPN-Skeptic for networks in Figure 3.2 ($p = 100$), sample sizes $n = 200$, 1000, 2000 and $\lambda_{noise} = 5$.

methods, and differences among algorithms are more evident. The TP score of PC-LPGM becomes already reasonable when n approaches 2000 observations, as it will be explained in Section 3.5. It is worth noting that performances of methods based on l_1-regularized regression are overall less accurate and more variable in this scenario. For example, the number of recovered edges with LPGM is almost comparable to an empty graph in a number of cases, a result possibly related to the levels of β chosen in the exercise. To ascertain such explanation, we run some simulations with higher variability threshold levels, i.e, 0.5 and 0.3 for LPGM (results not reported here). Although the TP scores improved, they were still unable to compete with the best performing algorithms.

Overall, results seem to demonstrate the good performances of PC-LPGM algorithm in all considered situations.

Table 3.1: Monte Carlo marginal means of TP, FP, FN, PPV, Se obtained by simulating 500 samples from each of the three networks shown in Figure 3.1 ($p = 10$).

λ_{noise}	n	Algorithm	TP	FP	FN	PPV	Se
0.5	200	PC-LPGM	6.336	0.067	2.023	0.991	0.758
		LPGM	3.768	0.255	4.565	0.955	0.449
		PDN	5.784	1.035	2.549	0.858	0.696
		VSL	4.169	0.033	4.190	0.995	0.498
		GLASSO	4.076	0.026	4.283	0.996	0.487
		NPN-Copula	4.568	0.029	3.791	0.996	0.546
		NPN-Skeptic	4.476	0.034	3.883	0.995	0.534
	1000	PC-LPGM	8.359	0.090	0.000	0.990	1.000
		LPGM	5.307	1.909	3.027	0.869	0.637
		PDN	5.991	0.721	2.342	0.901	0.722
		VSL	4.694	0.000	3.665	1.000	0.562
		GLASSO	4.624	0.000	3.735	1.000	0.554
		NPN-Copula	4.954	0.000	3.405	1.000	0.592
		NPN-Skeptic	4.819	0.000	3.540	1.000	0.576
	2000	PC-LPGMC	8.422	0.090	0.000	0.990	1.000
		LPGM	7.132	4.639	1.201	0.690	0.856
		PDN	5.981	0.694	2.353	0.904	0.721
		VSL	5.657	0.000	2.765	1.000	0.675
		GLASSO	5.620	0.000	2.802	1.000	0.670
		NPN-Copula	5.901	0.000	2.521	1.000	0.702
		NPN-Skeptic	5.779	0.000	2.643	1.000	0.688
ine	200	PC-LPGM	2.059	0.704	6.357	0.755	0.245
		LPGM	1.589	2.124	6.744	0.495	0.191
		PDN	3.465	4.660	4.869	0.435	0.415
		VSL	1.849	0.789	6.567	0.768	0.220
		GLASSO	1.834	0.787	6.582	0.768	0.218
		NPN-Copula	1.952	0.688	6.465	0.802	0.232
		NPN-Skeptic	1.768	0.726	6.648	0.775	0.210

Table 3.1 – continued from previous page

λ_{noise}	n	Algorithm	TP	FP	FN	PPV	Se
5	1000	PC-LPGM	7.889	1.063	0.444	0.890	0.946
		LPGM	4.115	2.176	4.219	0.686	0.494
		PDN	5.853	1.249	2.481	0.833	0.703
		VSL	3.135	0.012	5.198	0.998	0.377
		GLASSO	3.118	0.012	5.215	0.998	0.375
		NPN-Copula	3.211	0.006	5.122	0.999	0.386
		NPN-Skeptic	3.007	0.008	5.327	0.998	0.362
	2000	PC-LPGM	8.355	1.056	0.002	0.897	1.000
		LPGM	4.337	2.151	3.996	0.703	0.520
		PDN	6.153	0.805	2.180	0.892	0.740
		VSL	3.954	0.000	4.404	1.000	0.473
		GLASSO	3.931	0.000	4.426	1.000	0.470
		NPN-Copula	4.094	0.000	4.264	1.000	0.490
		NPN-Skeptic	3.863	0.000	4.494	1.000	0.462

TABLE 3.2: Monte Carlo marginal means of TP, FP, FN, PPV, Se obtained by simulating 500 samples from each of the three networks shown in Figure 3.2 ($p = 100$).

λ_{noise}	n	Algorithm	TP	FP	FN	PPV	Se
0.5	200	PC-LPGM	54.780	9.430	46.913	0.822	0.535
		LPGM	5.879	2.414	94.550	0.786	0.058
		PDN	41.476	47.640	59.524	0.493	0.406
		VSL	57.990	24.512	43.703	0.703	0.566
		GLASSO	56.531	23.983	45.161	0.703	0.552
		NPN-Copula	60.315	21.202	41.378	0.737	0.589
		NPN-Skeptic	58.967	26.466	42.726	0.695	0.576
	1000	PC-LPGM	98.693	13.201	4.398	0.882	0.956
		LPGM	34.694	0.377	66.152	0.929	0.339
		PDN	68.954	9.723	32.046	0.890	0.688
		VSL	81.930	0.107	21.160	0.999	0.782
		GLASSO	81.316	0.129	21.775	0.998	0.776
		NPN-Copula	85.150	0.078	17.941	0.999	0.814
		NPN-Skeptic	84.277	0.160	18.814	0.998	0.806
	2000	PC-LPGMC	101.508	14.405	1.114	0.879	0.990
		LPGM	43.743	0.305	57.257	0.872	0.421
		PDN	73.431	3.448	27.569	0.953	0.736
		VSL	93.355	0.004	9.266	1.000	0.904
		GLASSO	93.127	0.004	9.494	1.000	0.902
		NPN-Copula	96.317	0.000	6.305	1.000	0.935
		NPN-Skeptic	95.303	0.006	7.319	1.000	0.924
ine							
	200	PC-LPGM	6.170	14.292	94.830	0.288	0.060

Table 3.2 – continued from previous page

λ_{noise}	n	Algorithm	TP	FP	FN	PPV	Se
5		LPGM	7.075	56.433	93.925	0.124	0.068
		PDN	11.220	97.543	89.780	0.104	0.110
		VSL	7.752	23.011	93.248	0.276	0.076
		GLASSO	7.505	21.932	93.495	0.280	0.073
		NPN-Copula	8.156	21.971	92.844	0.297	0.079
		NPN-Skeptic	7.875	25.844	93.125	0.265	0.077
	1000	PC-LPGM	61.903	24.368	39.097	0.692	0.604
		LPGM	1.383	2.055	99.617	0.470	0.014
		PDN	43.153	49.657	57.847	0.488	0.423
		VSL	11.901	0.584	87.650	0.953	0.119
		GLASSO	11.913	0.583	87.638	0.953	0.119
		NPN-Copula	13.537	0.569	86.014	0.958	0.135
		NPN-Skeptic	13.036	0.765	86.515	0.945	0.130
	2000	PC-LPGM	88.478	26.908	12.522	0.761	0.871
		LPGM	1.801	1.368	99.199	0.548	0.018
		PDN	60.703	23.902	40.297	0.751	0.600
		VSL	21.141	0.017	80.179	0.999	0.206
		GLASSO	21.701	0.017	79.618	0.999	0.212
		NPN-Copula	26.557	0.011	74.763	0.999	0.260
		NPN-Skeptic	25.228	0.017	76.092	0.999	0.247

3.4 Real data analysis: inferring networks from next generation sequencing data

To make our evaluation of PC-LPGM stronger, we perform some biological validation by applying the new algorithm to level III breast cancer microRNAs (miRNAs) expression, retrieved from the Cancer Genome Atlas. Here, we expect to obtain results coherent with the current biological knowledge.

miRNAs are non-coding RNAs that are transcribed but do not encode proteins. miRNAs have been reported to play a pivotal role in regulating key biological processes, e.g., post-transcriptional modifications and translation processes. Some studies revealed that some disease-related miRNAs can indirectly regulate the function of other miRNAs associated with the same phenotype. In this perspective, studying the features of the interaction pattern of miRNAs in some conditions might help understand complex phenotype conditions.

Here, we consider level III breast cancer. Our interest lies in the pattern of interactions among miRNAs, with a particular focus on the existence of hubs. In fact, nodes with atypically high number of connections represent sites of signalling convergence with potentially large explanatory power for network behaviour or utility for clinical prognosis and therapy. By applying our algorithm, we expect to obtain results in line with known associations between miRNAs and breast cancer, and possibly gain more understanding of the nature of their effect on other genes. In other words, we expect some miRNAs associated with this phenotype to be the hubs of our estimated structure.

miRNAs expression, obtained by high-throughput sequencing, was downloaded from The Cancer Genome Atlas (TCGA) portal (`https://tcga-data.nci.nih.gov/docs/publications/brca_2012/`). The raw count data set consisted of 544 patients and 1046 miRNAs. As measurements were zero-inflated and highly skewed, with total count volumes depending on experimental condition, standard preprocessing was applied to the data (see Allen and Liu (2013)). In particular, we normalized the data by the 75% quantile matching [Bullard *et al.* (2010)]; selected top 25% most variable mirRNA across the data; used a power transform X^{α} for $\alpha \in [0, 1]$ with α chosen via the minimum Kolmogorov-Smirnov statistic [Li *et al.* (2012)]. The miRNAs with little variation across the samples were filtered out, leaving 544 patients and 261 miRNAs. The effect of preprocessing steps on four prototype miRNA are shown in Figure 3.9.

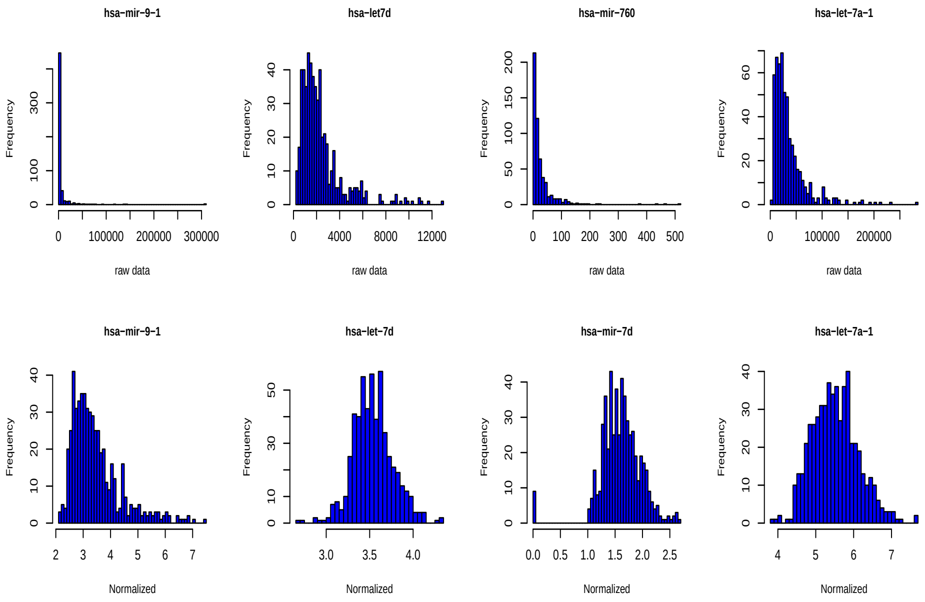


Figure 3.9: Distribution of four miRNA-Seq: raw data (top), normalized (bottom) data.

Normalized data was used as input to PC-LPGM. A significance level of 5% resulted in a sparse graph; the network resulting by fixing a significance level of 5% is shown in Figure 3.10.

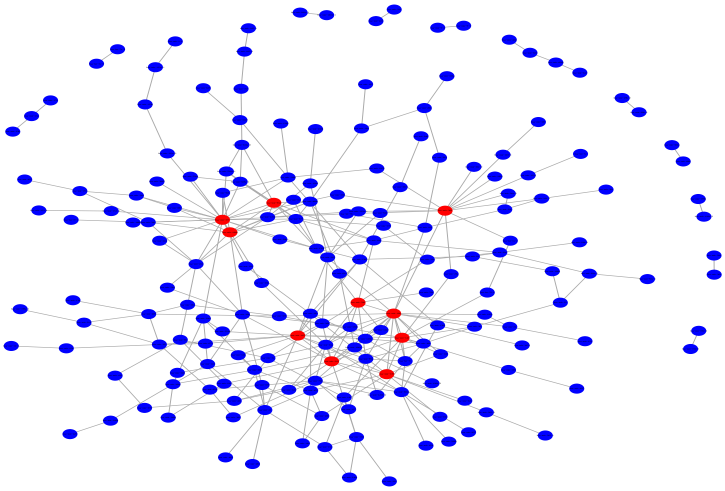


FIGURE 3.10: Breast cancer miRNA network estimated by the PC-LPGM algorithm (hub nodes coloured red).

We identified ten hub nodes, in the network, i.e., miR-10b, -30a, -143, -375, -145, -210, -139, -934, -190b, -590. Almost all of them are known to be related to breast cancer [Volinia *et al.* (2012)], providing a biological validation of the potential of the algorithm to recover the sites of the network with high explanatory power. In particular, miR-10b and -210 highly express in breast cancer, when high expression is related to poor prognosis; miR-30a, -143 and -145 appear to be inhibitors of progression, and should therefore be low in patients with good survival (Zhang *et al.* (2014), Yan *et al.* (2014)). These results play the role of a biological validation of the ability of PC-LPGM to retrieve structures reflecting existing relations among variables.

3.5 Discussion

The main contribution of this chapter is a careful analysis of the numerical and statistical efficiency of PC-LPGM, a simple method for structure learning of undirected graphical models for Poisson data. A key strategy of our approach is controlling the number of variables in the conditional sets, as done in the PC algorithm. In this way, we control problems of estimation when the number of random variables p is large.

Our main theoretical result provides sufficient conditions on the triple (n, p, m) and on the model parameters for the method to succeed in consistently estimating the neighbours of every node in the graph. Indeed, Theorem 3 in Peña *et al.* (2009) guarantees that in our setting, there exists a strictly positive distribution that is faithful to G, and Assumption 3.2.10 excludes, at least theoretically, from the parameter spaces the subspace on which the faithfulness condition is not satisfied. The possibility of possible violations of faithfulness in our setting is still under consideration. Moreover, Theorem 4.3.1 not only specifies sufficient conditions but it also provides the probability with which the method recovers the true edge set. Indeed, Equation (3.19) shows that

$$\mathbb{P}_{\boldsymbol{\theta}}(\text{ error occur in PC-procedure}) \leq O(p^{m+2})\big[\exp(-c_4 n^{1-\kappa_2}) + c_2\nu^{-5/4} + \exp(-c_3 n)\big],$$

where $\nu = \max\{n, p\}$. Hence, the right hand sight of the Equation will tend to 0 if $p^{m+2}\nu^{-5/4} \to 0$. As we are considering the case of a fixed number p of random variables, then $\nu = n$ when n is large enough. Thus, the sufficient condition becomes

$$\begin{aligned} p^{m+2}n^{-5/4} &\leq 1 \\ \Rightarrow (p^{m+2})^{4/5} &\leq n. \end{aligned} \tag{3.21}$$

In our simulation setting, we consider sparse graphs. Roughly speaking, the maximum number of neighbours m is 2 for $p = 10$, and $m = 1$ for $p = 100$. Hence, for $p = 10$, Equation (3.21) suggests that $10^{16/5} \approx 1000$ observations are enough to have consistency of the proposed algorithm. This result is confirmed by simulation results. Similarly, when $p = 100$ and $m = 1$ the number of observations needed to have convergence is $100^{12/5} \approx 10000$. This remark explains the reason why we did not get good results for the case of high dimensional setting, i.e., $p = 100$, and $n = 200, 1000, 2000$. However, it is worth to note that this bound is just a sufficient number.

Chapter 4

Guided structure learning of DAGs

As stated before, we are motivated by the need of defining a statistical framework for modelling the interactions between genes. As the interest usually lies in the direction of influence, directed graphs are usually preferred to the undirected graphs. DAGs are particularly convenient models to present such networks.

In this chapter, we tackle structure learning of DAGs, with the idea of exploiting available prior knowledge of the domain at hand to guide the search of the best structure. In particular, we will assume to know the topological ordering of variables in addition to the given data. Recall that a topological ordering of a directed graph is an ordering of its nodes s.t., for every directed edge $X_i \to X_j$, X_i precedes X_j in the ordering. The set of vertices that precedes X_j will be denoted by $pre(j)$.

Although the assumption of knowledge of the topological ordering of variables beforehand might restrict the potential of our proposals, it provides an opportunity to easily include prior knowledge leading to a significant improvement of the accuracy and a considerable reduction of computational costs as it considerably reduces the search space.

In detail, we propose three new algorithms based on a modification of: (i) a very popular structure learning algorithm, i.e., the K2 algorithm; (ii) the LPGM algorithm; (iii) the PC algorithm. Sections 4.1, 4.2, 4.3 are devoted to the three proposed methods, i.e., PK2, Or-LPGM, and Or-PPGM. In Section 4.4, we provide some experimental results that illustrate the performance of our methods in practice. Some conclusions and remarks are provided in Section 4.5.

We note that exploiting the topological ordering of the variables guarantees that conditions in order to prove faithfulness of the distribution to the DAG in this setting are satisfied (see Section 7.4 in Sadeghi (2017)), which allows us to prove consistency of our algorithms.

4.1 The PK2 algorithm

A popular guided algorithm for structure learning is the K2 algorithm, which assumes that data arise from a categorical random variable. A natural choice for using K2 on non-categorical data is categorization, a choice that can work well in some circumstances, but, unfortunately can also be ill-suited. In this section, we follow another line and extend the K2 algorithm to count data. Our proposal, named PK2 (Poisson K2), combines the assumption of Poisson node conditional distributions with the greedy search of the K2 algorithm.

To extend the K2 method to count data, we substitute the K2 score with an alternative scoring criterion respectful of the nature of the data. When considering alternative criteria, we restricted our attention to criteria that balance the goodness of fit and model parsimony.

The first choice is the Bayesian information criterion (BIC), introduced by Schwarz (1978), which is, in its most general form, given by

$$BIC = 2\ell(\hat{\boldsymbol{\theta}}, \mathbb{X}) - \log(n)(\text{number of parameters}).$$

The BIC is one of the most widely known and pervasively used tools in statistical model selection. Its popularity derives from its computational simplicity and effective performance in many modeling frameworks, including Bayesian applications where prior distributions may be elusive.

The second alternative model selection criterion is the Akaike information criterion (AIC), named after its inventor Akaike Hirotugu [Akaike (1974)]. While playing the same role as the BIC score, the AIC penalizes the number of parameters less strongly than the BIC does. In detail, the AIC score is expressed as follow:

$$AIC = 2\ell(\hat{\boldsymbol{\theta}}, \mathbb{X}) - k(\text{number of parameters}),$$

where the default value of k is 2 in the classical AIC.

Although the expressions of BIC and AIC look very similar, they originate from quite different frameworks. BIC assumes that the true model is included in the set of candidate models and measures the belief that a certain model is the true data generating model. In contrast, AIC does not assume that any of the candidate models is necessarily true, and calculates the Kullback-Leibler discrepancy, i.e., the distance between the probability density generated by the model and reality. Hence, a formal comparison in terms of performance between AIC and BIC is very difficult (see Burnham and Anderson (2003), and Wagenmakers and Farrell (2004)).

Once the score is specified, we employ the same search strategy as in the K2 algorithm. This guarantees that PK2 inherits the strengths of the original algorithm, ease of implementation, and feasibility up to hundreds of nodes.

Let $g(.)$ be the BIC or the AIC score. The pseudo code of the PK2 algorithm identical to that of K2, but using the difference score function $g(.)$ is given in Algorithm 2.

Algorithm 2 PK2 Algorithm.

```
Input $n$ independent realizations of the $p$-random vector $\mathbf{X}$, i.e., $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(n)}$;
  a topological ordering $Ord$ on nodes, an upper bound for the number $u$ of parents
  of a node may have.
for $i = Ord(1)$ to $Ord(p)$ do
    $pa_i \leftarrow \emptyset$
    $P_{old} \leftarrow g(i, pa_i)$
    OKToProceed $\leftarrow$ TRUE
    while OKToProceed and $|pa_i| < u$ do
        let $z$ be the node in $pre(x_i) \setminus pa_i$ that maximizes $g(i, pa_i \cup \{z\})$
        $P_{new} \leftarrow g(i, pa_i \cup \{z\})$
        if $P_{new} > P_{old}$ then
            $P_{old} \leftarrow P_{new}$
            $pa_i \leftarrow pa_i \cup \{z\}$
        else OKToProceed $\leftarrow$ FALSE
        end if
    end while
    print "parents of node" $x_i$, "are" $pa_i$.
end for
```

4.1.1 Asymptotic property

Properties of K2 guarantee that the algorithm identifies the true structure up to an equivalent class as the number of observations goes to infinity when a consistent scoring criterion is used. As both BIC and AIC are consistent scoring criteria [Haughton *et al.* (1988)], this property is inherited by PK2. Moreover, the Poisson model is identifiable (see Section 2.2). Therefore, the estimator in Algorithm 2 convergences asymptotically to the true graph.

4.2 The Or-LPGM algorithm

Our second proposal, i.e., Or-LPGM, is the natural extension to Poisson DAGs of the structure learning algorithm for Poisson undirected graphs, i.e., LPGM.

Let $i_1, i_2, \ldots, i_p$ be a topological ordering. The conditional distribution of each variable X_{i_s} given its precedents $pre(i_s)$ in the topological ordering $i_1, i_2, \ldots, i_p$ follows a Poisson distribution, i.e.,

$$X_{i_s}|\mathbf{x}_{pre(i_s)} \sim \text{Pois}(\exp\{\sum_{t \in pre(i_s)} \theta_{i_s t} x_t\}), \quad s \in V = \{1, \ldots, p\}. \tag{4.1}$$

Then, the node conditional distribution can be rewritten as

$$\begin{aligned} \mathbb{P}_{\boldsymbol{\theta}_{pre(i_s)}}(x_{i_s}|\mathbf{x}_{pre(i_s)}) &= \exp\left\{x_{i_s} \sum_{t=i_1}^{i_{(s-1)}} \theta_{i_s t} x_t - \log(x_{i_s}!) - e^{\sum_{t=i_1}^{i_{(s-1)}} \theta_{i_s t} x_t}\right\} \quad (4.2)\\ &= \exp\left\{x_{i_s} \langle \boldsymbol{\theta}_{pre(i_s)}, \mathbf{x}_{pre(i_s)}\rangle + C(x_{i_s}) - D(\langle \boldsymbol{\theta}_{pre(i_s)}, \mathbf{x}_{pre(i_s)}\rangle)\right\}. \end{aligned}$$

A rescaled negative node conditional log-likelihood can be written as follows

$$\begin{aligned} l(\boldsymbol{\theta}_{pre(i_s)}, \mathbb{X}_{i_s}; \mathbb{X}_{pre(i_s)}) &= -\frac{1}{n} \log \prod_{t=1}^{n} \mathbb{P}\boldsymbol{\theta}_{pre(i_s)}(x_{is}|\mathbf{x}^{(t)}_{pre(i_s)}) \quad (4.3)\\ &= \frac{1}{n} \sum_{t=1}^{n} \left[-x_{ti_s} \langle \boldsymbol{\theta}_{pre(i_s)}, \mathbf{x}^{(t)}_{pre(i_s)}\rangle + D(\langle \boldsymbol{\theta}_{pre(i_s)}, \mathbf{x}^{(t)}_{pre(i_s)}\rangle)\right]. \end{aligned}$$

Structure learning can be performed by mean of p-local regressions aimed at identifying the set of non-zero parameters $\theta_{i_s t}$. In this case, the parameter $\boldsymbol{\theta}_{pre(i_s)}$ is estimated by minimizing the rescaled negative node conditional log-likelihood (4.3), i.e.,

$$\hat{\boldsymbol{\theta}}_{pre(i_s)} = \text{argmin}_{\boldsymbol{\theta}_{pre(i_s)} \in \mathbb{R}^{s-1}} l(\boldsymbol{\theta}_{pre(i_s)}, \mathbb{X}_{i_s}; \mathbb{X}_{pre(i_s)}). \tag{4.4}$$

To encourage sparsity of estimated graphs, a l_1- regularized conditional log-likelihood can be considered, i.e.,

$$\hat{\boldsymbol{\theta}}_{pre(i_s)} = \text{argmin}_{\boldsymbol{\theta}_{pre(i_s)} \in \mathbb{R}^{s-1}} l(\boldsymbol{\theta}_{pre(i_s)}, \mathbb{X}_{i_s}; \mathbb{X}_{pre(i_s)}) - \lambda \|\boldsymbol{\theta}_{pre(i_s)}\|_1.$$

Given the solution $\hat{\boldsymbol{\theta}}_{pre(i_s)}$, the set of parents of node s is given by

$$\hat{pa}(s) = \{t \in pre(i_s) : \quad \hat{\theta}_{st} \neq 0\}.$$

In summary, the Or-LPGM algorithm takes the topological ordering as a prior knowledge to restrict the set of candidate parents. Then, the structure learning process boils down to p (possibly penalized) standard regressions, where the regularization parameter λ, that controls the sparsity of the graph structure, is chosen by the cross validation method.

Let k be the number of folds, and S_i is the training set of i-th fold. The pseudo code of the Or-LPGM algorithm is given in Algorithm 3.

Algorithm 3 Or-LPGM Algorithm.

```
Input $n$ independent realizations of the $p$-random vector $\mathbf{X}$, i.e., $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}$;
    a topological ordering on nodes.
Generate a range value of $\lambda$, $\Lambda = \{\lambda_{max}, \ldots, \lambda_{min}\}$
for $i = 1$ to $k$ do in parallel
    for $j = 1$ to $p$ do
        fitting a $l_1$- penalized log-linear regression to $X_j$, path-wise for $\Lambda$ on
            sub-data $S_i$.
    end for
end for
Determine $\lambda_{opt}$ via cross validation method.
Return the optimal graph, $G_{\lambda opt}$
```

4.2.1 Consistency of the Or-LPGM algorithm

Consistency of Or-LPGM can be proved easily, as shown in the following theorem.

Theorem 4.2.1. Assume 3.2.1- 3.2.5. Denote by $\hat{G}(\lambda_n)$ the estimator resulting from Or-LPGM, and by G the true graph. Then, there exists a numerical sequence λ_n, s.t.

$$\mathbb{P}_{\boldsymbol{\theta}}(\hat{G}(\lambda_n) = G) = 1,$$

when $n \longrightarrow \infty$.

Proof. For a fixed topological ordering, Theorem 4.2.1 is proved by following the lines of Corollary 7 in Yang *et al.* (2015). We note that the set of explanatory variables $\mathbf{X}_{pre(i_s)}$ in the generalized linear model X_s given $\mathbf{X}_{pre(i_s)}$ does not include variable X_t, $t \in \{V \backslash \{s\}\} \backslash pre(s)$. Suppose we zero-pad the true parameter $\boldsymbol{\theta}^*_{pre(s)} \in \mathbb{R}^{|pre(s)|}$ to include zero weights over $\{V \backslash \{s\}\} \backslash pre(s)$, then the resulting parameter would lie in $\mathbb{R}^{(p-1)}$.

It is worth to note that this proof does not depend on the topological ordering since identifiability of Poisson models guarantees that there is only one distribution associated to the true graph G. □

4.3 The Or-PPGM algorithm

Our third proposal, i.e., Or-PPGM, is based on a modification of the PC algorithm. As stated before, the PC algorithm can be combined with any consistent statistical test of conditional independence. In the Poisson case, conditional independences can be inferred from Wald type tests on the parameters θ_{st} (see Section 3.1). Therefore, one solution for recovering the underlying structure is performing conditional independence tests as in the PC-LPGM algorithm (see Section 3.1).

The algorithm starts from a complete DAG, i.e., a DAG obtained from a complete undirected graph by putting the direction for all edges following the topological ordering. At each level of the cardinality of the conditioning variable set $\mathbf{S}$, we test, at some pre-specified significance level, the null hypothesis $H_0 : \theta_{st|\mathbf{K}} = 0$, with $\mathbf{K} = \mathbf{S} \cup \{t\}$. If the null hypothesis is not rejected, the edge $t \to s$ is considered to be absent from the graph. We note that the cardinality of the set $\mathbf{S}$ increases from 0 to $\min\{ord(s)-1, m\}$, where $ord(s)$ is the position of X_s in the topological ordering, and $m < (p-2)$ is the maximum number of neighbours that one node is allowed to have. For a description of the conditional independence test, as well as the definition of Z-statistic, we refer readers to Section 3.1.

Assuming that the order of variables is specified beforehand considerably reduces the number of conditional independence tests. For example, for each $s \in V$, we test if data support existence of the conditional independence relation $X_s \perp\!\!\!\perp X_t | \mathbf{X_S}$ only for $t \in pre(s)$ and for any $\mathbf{S} \subset \{\{1, \ldots, p\} \cap pre(s)\} \backslash \{s, t\}$.

The pseudo-code of the Or-PPGM algorithm is given in Algorithm 4.

4.3.1 Consistency of the Or-PPGM algorithm

Consistency of Or-PPGM can be proved easily, as shown in the following theorem.

Theorem 4.3.1. Assume 3.2.1- 3.2.5. Denote by $\hat{G}(\alpha_n)$ the estimator resulting from Or-PPGM, and by G the true graph. Then, there exists a numerical sequence $\alpha_n \longrightarrow 0$, s.t.

$$\mathbb{P}_{\boldsymbol{\theta}^*}(\hat{G}(\alpha_n) = G) = 1,$$

when $n \longrightarrow \infty$.

Proof. For different topological orderings $T_1, T_2, \ldots, T_k$, Algorithm 4 performs sequences of tests $S_1, S_2, \ldots, S_k$, respectively. We note that $S_j,\ j = 1, \ldots k$ is a subsequence of the sequence of tests performed in PC-LPGM (Section 3.1). Hence, Theorem 4.3.1 shows that there exists a numerical sequence $\alpha_n \to 0$, s.t. the estimators $\hat{G}^T(\alpha_n),\ T = T_1, T_2, \ldots T_k$ convergence to the unique true graph (as the Poisson model is identifiable, see Section 2.2). □

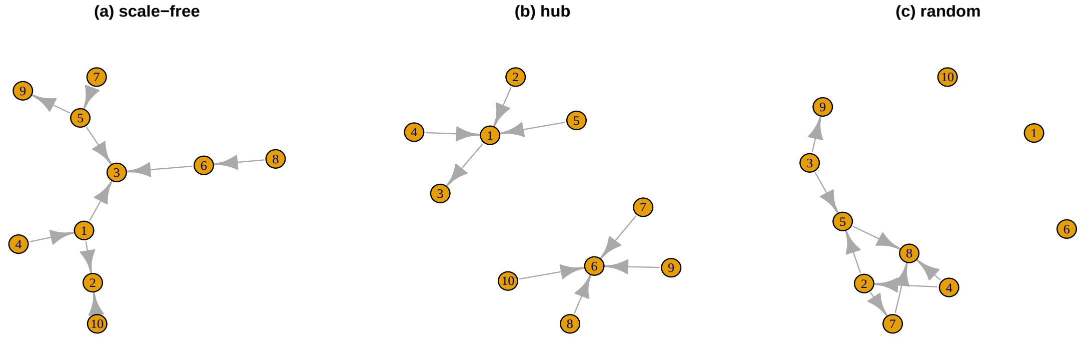
Algorithm 4 The Or-PPGM algorithm.

```
Input: $n$ independent realizations of the $p$-random vector $\mathbf{X}$, i.e., $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(n)}$;
  a topological ordering $Ord$, (and a stopping level $m$).
Output: An estimated DAG $\hat{G}$.
Form the complete undirected graph $\tilde{G}$ on the vertex set $V$.
Orient edges on $\tilde{G}$ to form DAG $G'$.
$l = -1; \quad \hat{G} = G'$
repeat
  $l = l + 1$
  for all vertices $s \in V$, do
    let $\mathbf{K}_s = pa(s)$
  end for
  repeat
    Select a (new) ordered pair of nodes $s, t$ that are adjacent in $\hat{G}$ s.t.
    $|\mathbf{K}_s \backslash \{t\}| \geq l$.
    repeat
      choose a (new) set $\mathbf{S} \subset \mathbf{K}_s \backslash \{t\}$ with $|\mathbf{S}| = l$.
      if $H_0 : \theta_{st|\mathbf{S}} = 0$ not rejected
        delete edge $(s, t)$ from $\hat{G}$
      end if
    until edge $(s, t)$ is deleted or all $\mathbf{S} \subset \mathbf{K}_s \backslash \{t\}$ with $|\mathbf{S}| = l$ have been considered.
  until all ordered pair of adjacent variables $s$ and $t$ such that $|\mathbf{K}_s \backslash \{t\}| \geq l$ and
    $\mathbf{S} \subset \mathbf{K}_s \backslash \{t\}$ with $|\mathbf{S}| = l$ have been tested for conditional independence.
until $l = m$ or for each ordered pair of adjacent nodes $s, t$: $|\text{adj}(\hat{G}, s) \backslash \{t\}| < l$.
```

4.4 Empirical study

Here, we empirically evaluate the ability of our proposals to retrieve the true DAG and compare them to a number of popular competitors. We again use PPV and Se (see Section 3.3) to evaluate their ability to reconstruct the true DAGs. As competitors, we consider structure learning algorithms for both Poisson and non Poisson variables. In detail, we consider PDN and ODS (see Section 2.3) as representatives of algorithms for Poisson data. To apply algorithms for categorical data, we categorize our data using two strategies, i.e., Gaussian mixture models on log transformed data shifted by 1 [Fraley and Raftery (2002)] and cutting points on the original scale of the data, where the range of the data is divided into n_{cut} pieces of equal length. To apply algorithms for continuous data, we log transform the data shifted by 1.

We consider the same cardinalities ($p = 10, p = 100$) and the same structures as in Chapter 3, i.e., (i) a scale-free graph; (ii) a hub graph; (iii) a random graph. To convert them into DAGs, we fix a topological ordering for each graph by taking a permutation of considered variables. Once the order is defined, undirected edges are oriented to form a DAG. See Figure 4.1 and 4.2 for a plot of the three chosen DAGs for $p = 10$ and $p = 100$, respectively.


(a) scale−free

(b) hub

(c) random

FIGURE 4.1: The graph structures for $p = 10$ employed in the simulation studies: (a) scale-free; (b) hub; (c) random graph.

Data generation

In order to simulate data, we first construct an adjacency matrix $Adj = (\theta_{ij})$ as follows:

1. fill in the adjacency matrix Adj with zeros;
2. replace every entry corresponding to a directed edge by one;
3. replace each entry equal to 1 with an independent realization from a Uniform($[-0.5, 0.5]$) random variable, representing the true values of parameter θ_{st}.

This yields a matrix Adj whose entries are either zeros or in the range $[-0.5, 0.5]$, representing positive and negative relations among variables. For each DAG corresponding to an adjacency matrix Adj, 500 datasets are sampled for three sample sizes, i.e., $n = 200, 1000, 2000$ as follows. The realization of the first random variable $X_{(i_1)}$ in the topological ordering $i_1, i_2, \ldots, i_p$ is sampled from a Pois($\exp(\theta_1)$), where the default value of θ_1 is 0. Realizations of the following random variables are recursively sampled from

$$X_{i_j}^{(t)} \sim \text{Pois}(\exp\{\sum_{k=i_1}^{i_{(j-1)}} \theta_{i_j k} x_{tk}\}).$$

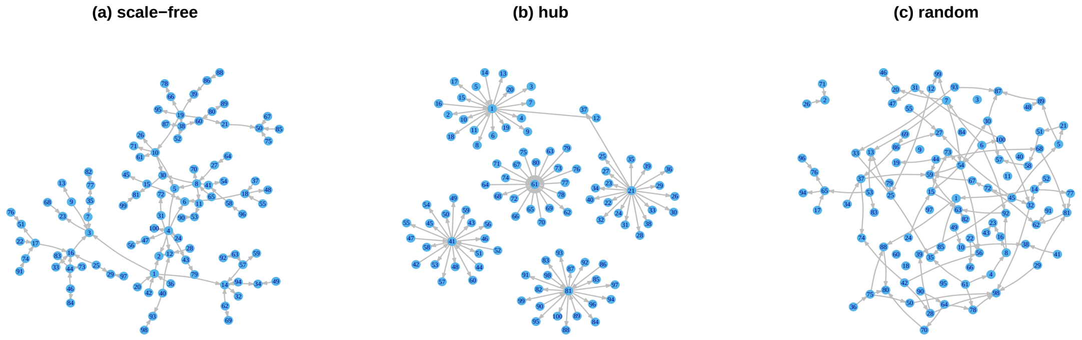


FIGURE 4.2: The graph structures for $p = 100$ employed in the simulation studies: (a) scale-free; (b) hub; (c) random graph.

Learning algorithms

The considered algorithms are listed below, along with specifications, if needed, of tuning parameters. In this study, beside the topological ordering, we also specify an additional input, i.e., the upper limit for the set of parents, m, which in this study was set to $m = 5$ for $p = 10$ and $m = 3$ for $p = 100$, respectively.

- **PKBIC**: PK2 using BIC;
- **PKAIC**: PK2 using AIC;
- **Or-PPGM**: level of significance of tests 1%;
- **Or-LPGM**: λ chosen via k-fold cross validation ($k = 10$);
- **PDN**;
- **ODS**: k-fold cross validation ($k = 10$);
- **K2mix**: K2 algorithm applied to data categorized by mixture models;
- **K2cut**: K2 algorithm applied to data categorized by $n_{cut} = 3$ cutting points;
- **MMHC**: Max Min Hill Climbing algorithm applied to data categorized by mixture models, using χ^2 tests of independence at the 1% significance level;
- **PCmix**: PC algorithm applied to data categorized by mixture models, using χ^2 tests of independence at the 1% significance level;

- **PClog**: PC algorithm applied to log transformed data, using Gaussian conditional independent tests at the 1% significance level.

We note that ODS, PDN and MMHC employ a preliminary step aimed to estimate the topological ordering. This makes the comparison with our algorithms not completely fair. Nevertheless, we decided to consider these algorithms in our numerical studies to get a measure of impact of the knowledge of the true topological ordering.

It is also worth to note that the PC algorithm returns a PDAGs that consists of both directed and undirected edges. In this case, we borrow the idea of Dor and Tarsi (1992) to extend a PDAG to DAG. For details of the algorithm, we refer to Section 5.1 or to the paper by Dor and Tarsi (1992).

Results

For the two considered vertex cardinalities, i.e., $p = 10, 100$, and for the chosen sample sizes, i.e., $n = 200, 1000, 2000$, Table 4.1 and Table 4.2 report, respectively, Monte Carlo means of TP, FP, FN, PPV and Se for each of considered method. Each value is computed as an average of the 1500 values obtained by simulating 500 samples for each of the three networks. Results disaggregated by network types are given in Appendix B, Tables B.5, and Tables B.6. These results indicate that the three proposed algorithms outperform, on average, Gaussian-based competitors (PClog), category-based competitors (K2, PCmix, MMHC), as well as the state-of-the-art algorithms that are specifically designed for Poisson graphical models (ODS, PDN) for $p = 10$, and they are competitive to the other approaches when $p = 100$.

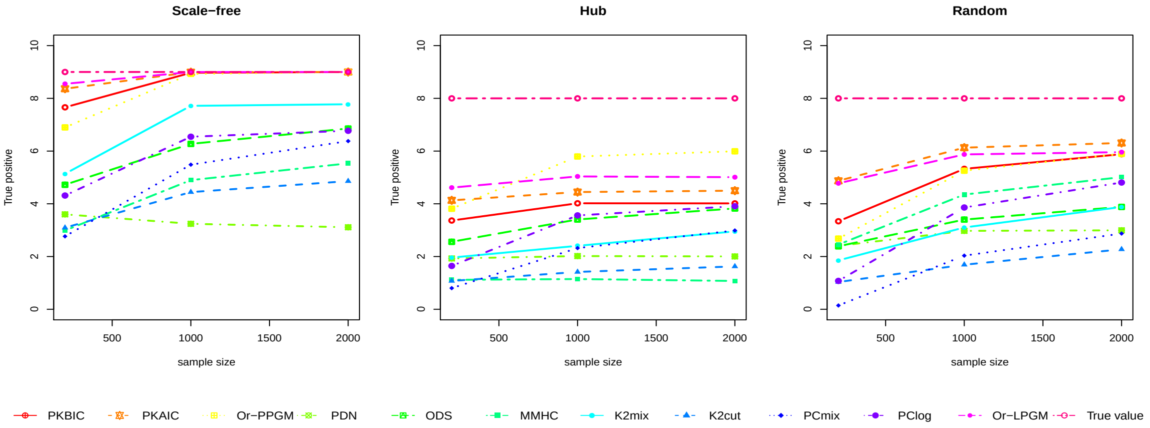


FIGURE 4.3: Number of TP edges recovered by PKBIC; PKAIC; Or-PPGM; Or-LPGM; PDN; ODS; MMHC; K2mix; K2cut; PCmix; PClog for networks in Figure 4.1 ($p = 10$) and sample sizes $n = 200$, 1000, 2000.

When $p = 10$, the proposed algorithms reach the highest TP value, followed by the ODS, the K2mix and the PClog algorithms. A closer look at the PPV and Se plot (see Figure 4.3) provides further insight into the behaviour of considered methods. The PK2BIC, the Or-LPGM and the Or-PPGM algorithm always reach the highest PPV and Se, while the PK2AIC usually have smaller PPV. This result is not surprising since the PK2AIC using the AIC criterion penalizes less strongly than the BIC one. As a consequence, the PK2AIC results in graphs with additional edges. This result is expected as we are considering the case $p = 10$, i.e., a relatively low dimensional model, in which the BIC is known to outperform AIC [Wagenmakers and Farrell (2004)].

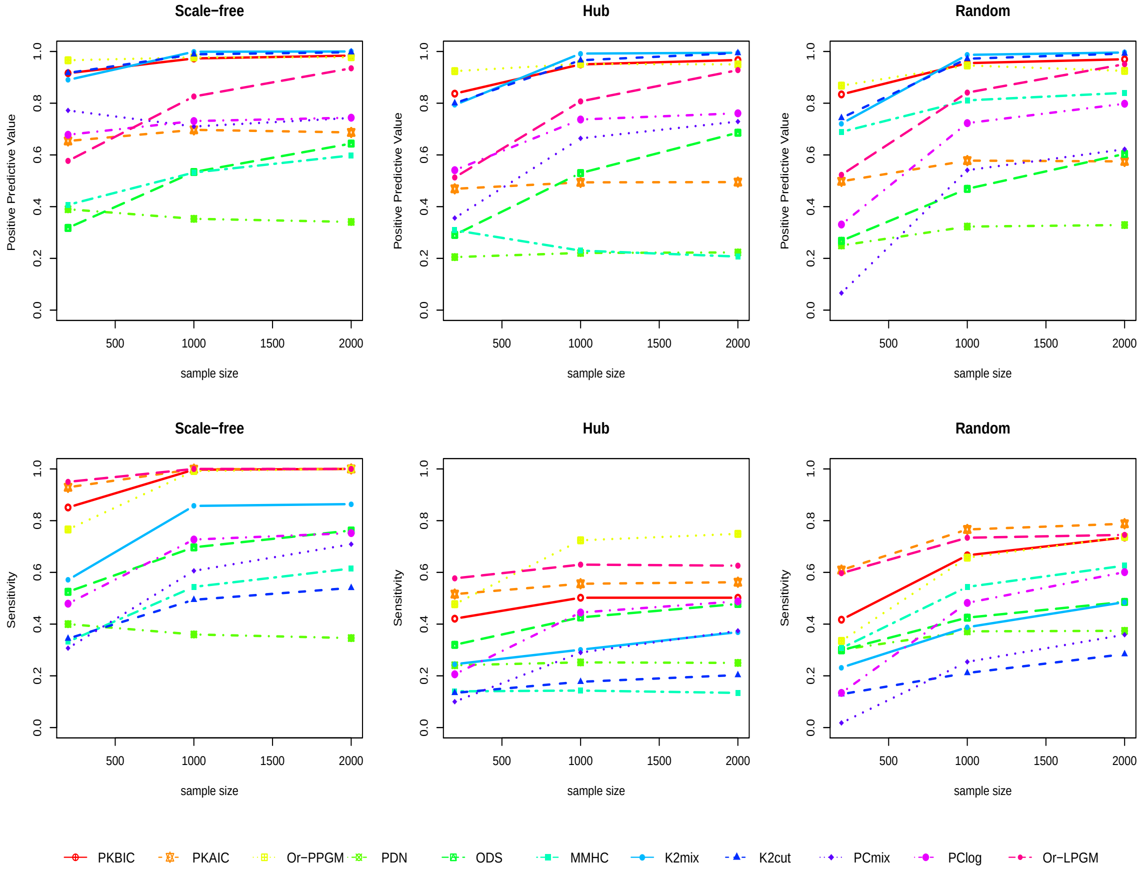


FIGURE 4.4: PPV (first panel row) and Se (second panel row) for PKBIC; PKAIC; Or-PPGM; Or-LPGM; PDN; ODS; MMHC; K2mix; K2cut; PCmix; PClog for networks in Figure 4.1 ($p = 10$), sample sizes 200, 1000, 2000.

It is interesting to note that the performance of PK2 and Or-LPGM appears to be better than that of the competing algorithms employing the Poisson assumption, i.e., PDN and ODS. The use of the topological ordering overcomes the inaccuracies of the first step of the ODS algorithm, i.e., identification of the order of variables, as well as the uncertainties in recovering the direction of interactions in PDN. On the other side, we also need to stress the good performances of Or-PPGM related to the difference between penalization and restriction of the conditional sets. In the PDN algorithm, as well as in the ODS algorithm, a prediction model is fitted locally on all other variables, by mean of a series of independent penalized regressions. In contrast, Or-PPGM controls the number of variables in the conditional sets for node s, which is progressively increased from 0 to $\min\{m, ord(s) - 1\}$.

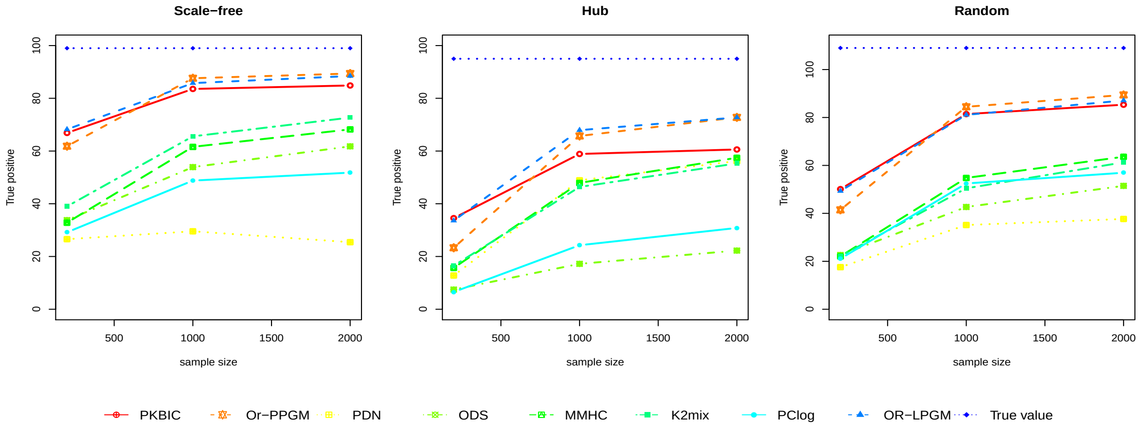


FIGURE 4.5: Number of TP edges recovered by PKBIC; Or-PPGM; Or-LPGM; PDN; ODS; MMHC; K2mix; PClog for networks in Figure 4.2 ($p = 100$) and sample sizes $n = 200$, 1000, 2000.

When considering other methods, i.e., category based methods (K2mix, K2cut, PCmix, MMHC), and Gaussian based method (PClog), we were surprised by the behaviour of the two best algorithms, i.e., K2mix, and PClog. Results show that with the same testing procedure, log transforming the data is better than categorizing data; and with the same greedy search strategy, data categorizing using mixture models is better than data categorizing using cutting points. On the basis of these results, we decided not to perform PCmix and K2cut in the case $p = 100$.

Results for the high dimensional setting ($p = 100$) are somehow comparable to the ones of the previous setting, as it can be seen in Figures 4.5, and 4.6. PK2BIC, Or-PPGM, and Or-LPGM still rank as the top three best algorithms, but differences among algorithms are more evident.

As a final remark, we note that the performances of ODS, PDN and MMHC are overall less accurate and more variable, as expected. This empirically confirms the relevant role played by the use of the topological ordering in terms of PPV and TP.

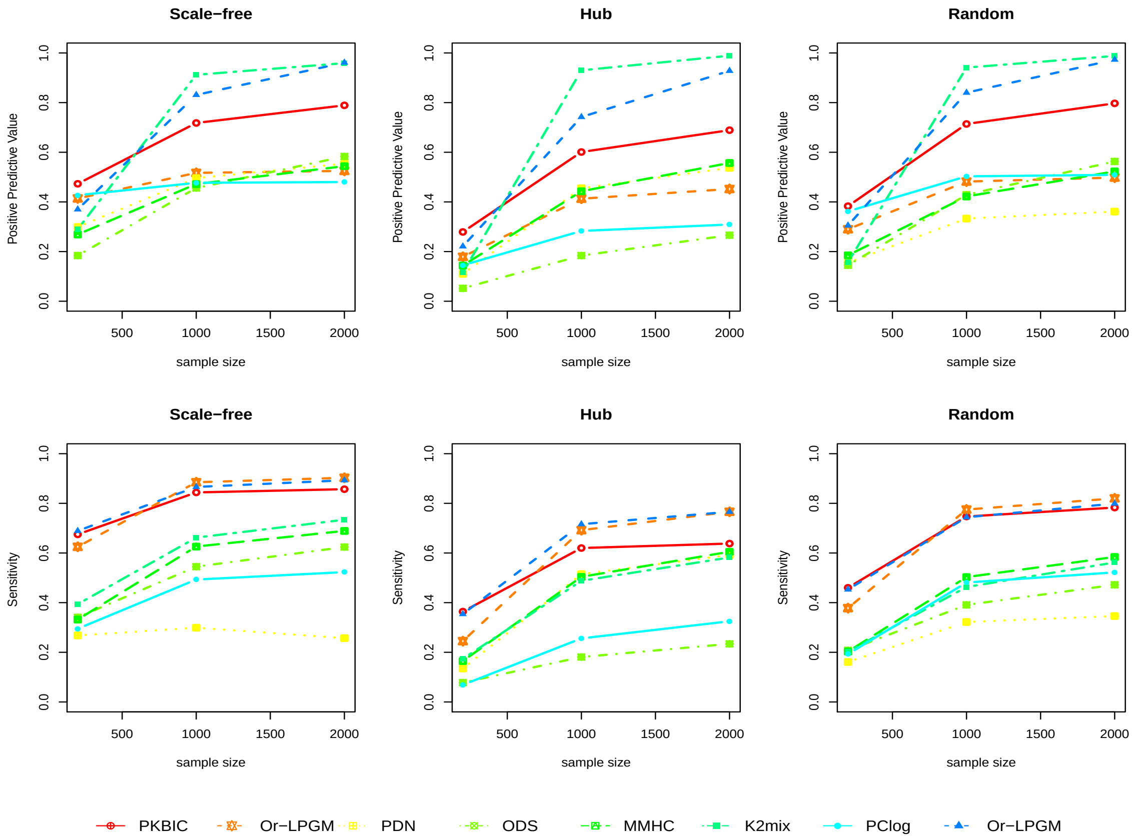


FIGURE 4.6: PPV (first panel row) and Se (second panel row) for PKBIC; Or-PPGM; Or-LPGM; PDN; ODS; MMHC; K2mix; PClog for networks in Figure 4.2 ($p = 100$), sample sizes n = 200, 1000, 2000.

TABLE 4.1: Monte Carlo marginal means of TP, FP, FN, PPV, Se obtained by simulating 500 samples from each of the three networks shown in Figure 4.1 ($p = 10$).

λ_{noise}	n	Algorithm	TP	FP	FN	PPV	Se
	200	PKBIC	4.788	0.779	3.545	0.862	0.563
		PKAIC	5.789	4.976	2.544	0.541	0.685
		Or-PPGM	4.464	0.385	3.869	0.920	0.526
		Or-LPGM	5.981	5.652	2.353	0.538	0.708
		PDN	2.645	6.805	5.689	0.282	0.314
		ODS	3.225	8.285	5.109	0.293	0.381
		MMHC	2.186	2.822	6.147	0.468	0.259
		K2mix	2.984	0.801	5.349	0.802	0.349
		K2cut	1.734	0.433	6.599	0.822	0.202
		PCmix	1.237	1.161	7.096	0.409	0.142
		PClog	2.345	1.703	5.989	0.517	0.273
	1000	PKAIC	6.105	0.275	2.228	0.959	0.722
		PKBIC	6.523	4.642	1.811	0.590	0.774
		Or-PPGM	6.660	0.314	1.673	0.958	0.791
		Or-LPGM	6.637	1.613	1.697	0.825	0.788

Table 4.1 – continued from previous page

λ_{noise}	n	Algorithm	TP	FP	FN	PPV	Se
		PDN	2.745	6.449	5.589	0.299	0.328
		ODS	4.361	4.386	3.973	0.511	0.516
		MMHC	3.465	3.089	4.868	0.524	0.410
		K2mix	4.408	0.031	3.925	0.993	0.515
		K2cut	2.517	0.071	5.817	0.976	0.294
		PCmix	3.271	1.659	5.063	0.638	0.384
		PClog	4.655	1.711	3.679	0.730	0.552
	2000	PKBIC	6.299	0.183	2.034	0.973	0.746
		PKAIC	6.602	4.757	1.731	0.586	0.784
		Or-PPGM	6.961	0.385	1.372	0.951	0.829
		Or-LPGM	6.657	0.503	1.677	0.938	0.790
		PDN	2.703	6.397	5.631	0.298	0.323
		ODS	4.855	2.830	3.479	0.644	0.575
		MMHC	3.873	2.962	4.460	0.548	0.459
		K2mix	4.870	0.015	3.463	0.997	0.573
		K2cut	2.920	0.019	5.413	0.994	0.342
		PCmix	4.079	1.655	4.254	0.698	0.480
		PClog	5.162	1.619	3.171	0.767	0.614

TABLE 4.2: Monte Carlo marginal means of TP, FP, FN, PPV, Se obtained by simulating 500 samples from each of the three networks shown in Figure 4.2 ($p = 100$).

λ_{noise}	n	Algorithm	TP	FP	FN	PPV	Se
	200	PKBIC	50.513	81.845	50.487	0.378	0.500
		Or-PPGM	42.133	98.485	58.867	0.294	0.416
		Or-LPGM	50.379	117.833	50.621	0.299	0.498
		PDN	18.969	90.596	82.031	0.188	0.188
		ODS	21.289	142.525	79.711	0.127	0.209
		MMHC	23.533	93.470	77.467	0.199	0.233
		K2mix	25.881	113.677	75.119	0.188	0.257
		PClog	18.936	38.365	82.064	0.311	0.186
	1000	PKBIC	74.626	35.098	26.374	0.678	0.737
		Or-PPGM	79.262	88.834	21.738	0.471	0.784
		Or-LPGM	78.266	19.186	22.739	0.804	0.775
		PDN	37.865	56.137	63.135	0.428	0.379
		ODS	37.931	66.563	63.069	0.356	0.372
		MMHC	54.885	68.566	46.115	0.446	0.544
		K2mix	54.138	4.415	46.862	0.928	0.538
		PClog	41.847	55.726	59.153	0.421	0.410
	2000	PKBIC	76.930	24.143	24.070	0.758	0.759
		Or-PPGM	83.861	86.378	17.007	0.491	0.830
		Or-LPGM	82.745	3.945	18.260	0.954	0.819
		PDN	39.765	47.931	61.235	0.484	0.398
		ODS	45.166	49.116	55.834	0.471	0.443

Table 4.2 – continued from previous page

λ_{noise}	n	Algorithm	TP	FP	FN	PPV	Se
		MMHC	63.094	53.902	37.906	0.541	0.626
		K2mix	63.095	1.541	37.905	0.979	0.626
		PClog	46.531	60.152	54.469	0.433	0.457

4.5 Conclusions and remarks

We have proposed three guided structure learning algorithms and compared them to a number of different approaches. Following this comparison, it appears that K2BIC, Or-LPGM, and Or-PPGM are promising algorithms in terms of prediction accuracy.

It is worth to note that comparison with algorithms designed for categorical or continuous data are obviously sensitive to the procedure of data transformation. We have noticed that the mixture based categorization is preferred to the simpler cut-points based categorization when applying the K2 algorithm; and making the data continuous by log transformation is better than categorizing the data when applying the PC algorithm. This is an important empirical conclusion that we draw from this study.

Finally, all proposed algorithms assume knowledge of the ordering of variables. This key ingredient makes the new algorithms more efficient, simple, and easy to implement. However, it also restricts their applications. Obviously, in many real life problems this knowledge is not available, or the topological ordering may be misspecified, or only a partial order on the set of nodes is specified due to a number of reasons. This difficulty motivated us to the next topic, i.e., unguided structure learning.

Chapter 5

Unguided structure learning of DAGs

In the previous chapter, our primary goal was to develop a framework to infer the underlying DAG from a set of given data. The three new algorithms for guided structure learning of DAGs, that we presented, answered this question when the topological ordering of the graph is assumed to be available. But, in many real situations, we might not know the topological ordering or it could be only unprecisely known. For example, when dealing with biological networks, the topological ordering may be misspecified due to inaccuracy of pathway representation or to the choices made in translating a pathway diagram into a fully directed graph.

Here, we turn our attention to unguided structure learning. We develop a new algorithm for learning DAGs which does not require prior knowledge of the ordering of variables. The structure of this chapter is as follows. In Section 5.1, we present our algorithm, learnDAG. Some empirical studies on ability of the algorithm are presented in Section 5.2. Some discussion is provided in Section 5.3.

5.1 The learnDAG algorithm

We slightly rewrite the joint distribution in (2.3) as

$$\begin{aligned}\mathbb{P}_{\boldsymbol{\theta}}(\mathbf{x}) &= \exp\Big\{\sum_{j=1}^{p}\theta_j x_j + \sum_{(k,j)\in E}\theta_{jk}x_j x_k - \sum_{j=1}^{p}\log(x_j!) - \sum_{j=1}^{p} e^{\theta_j+\sum_{k\in pa(j)}\theta_{jk}x_k}\Big\} \\ &= \exp\Big\{\sum_{j=1}^{p}\theta_j x_j + \sum_{j=1}^{p}\sum_{k\neq j}\theta_{jk}x_j x_k - \sum_{j=1}^{p}\log(x_j!) - \sum_{j=1}^{p} e^{\theta_j+\sum_{k\neq j}\theta_{jk}x_k}\Big\},\end{aligned}$$

where $\theta_{jk} = 0$ if $k \notin pa(j)$. Let $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(n)}$ be n samples independently drawn from the random vector $\mathbf{X}$, with $\mathbf{x}^{(i)} = (x_{i1}, x_{i2}, \dots, x_{ip})$, $i = 1, \dots, n$. Then, the log-likelihood function is of the form:

$$\begin{aligned} \ell(\boldsymbol{\theta}, \mathbb{X}) &= \sum_{i=1}^{n} \left\{ \sum_{j=1}^{p} \theta_j x_{ij} + \sum_{j=1}^{p} \sum_{k \neq j} \theta_{jk} x_{ij} x_{ik} - \sum_{j=1}^{p} \log(x_{ij}!) - \sum_{j=1}^{p} e^{\theta_j + \sum_{k \neq j} \theta_{jk} x_{ik}} \right\} \\ &= \sum_{j=1}^{p} \ell_j(\boldsymbol{\theta}_{V \setminus \{j\}}, \mathbf{x}_{V \setminus \{j\}}), \end{aligned} \tag{5.1}$$

where $\ell_j(\boldsymbol{\theta}_{V \setminus \{j\}}, \mathbf{x}_{V \setminus \{j\}}) = \sum_{i=1}^{n} \left\{ (\theta_j + \sum_{k \neq j} \theta_{jk} x_{ik}) x_{ij} - \log(x_{ij}!) - e^{\theta_j + \sum_{k \neq j} \theta_{jk} x_{ik}} \right\}$, denotes the node conditional log-likelihood for $X_j | \mathbf{x}_{V \setminus j}$. Learning DAGs can be performed by estimating parameters θ_{jk} that maximize the log-likelihood function $\ell(\boldsymbol{\theta}, \mathbb{X})$.

As seen in Chapter 3, without constraints on direction of edges, this problem can be solved by a number of algorithms, for example Yang *et al.* (2012), Allen and Liu (2013), Gallopin *et al.* (2013), Schelldorfer *et al.* (2014). However, the problem is extremely complicated by adding the constraint. The learning process is often divided in two steps: first, the topological ordering is retrieved, then parent sets are estimated. As already seen, Park and Raskutti (2015) proposed to recover the ordering by using an overdispersion score that measures the difference between the conditional mean and variance. However, this approach generally requires a large number of observations.

Here, we propose a new algorithm, called learnDAG. It consists of three main steps: 1) preliminary neighbourhood selection using existing methods for learning undirected graphs; 2) estimation of candidate parent sets using a log-likelihood score (or a BIC score); and 3) pruning of the DAG using variable selection algorithms such as Lasso, or significance tests. The pseudo code of the algorithm for a generic log-likelihood score is presented in Algorithm 5.

5.1.1 Step 1: preliminary neighbourhood selection (PNS)

The main purpose of Step 1 is to reduce computational complexity by imposing sparsity on the graph. Indeed, a candidate set of neighbours for each node could potentially include all nodes. Hence, learnDAG tries to reduce the dimensionality of the candidate set of neighbours for each node by preliminary estimating an undirected structure (see Figure 5.1). Any structure learning algorithm for undirected graphs could work to this aim, see for example Gallopin *et al.* (2013), Zhang and Mallick (2013), Yang *et al.* (2014), Žitnik and Zupan (2015). In what follows, we employ LPGM as in Allen and Liu (2013), and our proposals, PC-LPGM, presented in Section 3.1.

Algorithm 5 learnDAG algorithm.

```
Input: Data containing $n$ independent samples of the $p$-random vector $\mathbf{X}$; $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}$; (and an upper bound npa on the number of parents that a node may have).
Ouput: An estimated DAG
  Step 1: Estimate the undirected graph underlying data, construct neighbor sets $N(j)$.
  Step 2: Estimate potential parent set for each node,
    for i in 1: B do
      Perform Algorithm 7.
    end for
    Construct potential parent set for each node
  Step 3: Pruning the estimated DAG
```

When an estimated undirected graph is obtained, a neighbourhood set for each node j, denoted as $N(j)$, is built straightforwardly, that consists of all nodes potentially connected to node j. This simple step considerably reduces the overall computational complexity and running time, especially with large DAGs, and makes the algorithm feasible up to high-dimensional DAGs.

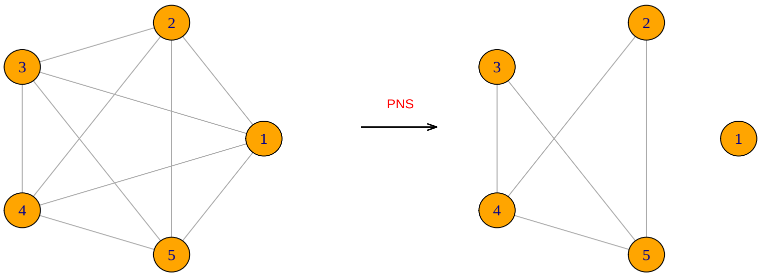


FIGURE 5.1: An example of applying the PNS step on an undirected graph consisting 6 nodes.

5.1.2 Step 2: estimating parent sets

We consider two strategies to estimate the candidate parent sets: (i) orientation rules, and (ii) a greedy search.

With the first strategy, the result of Step 1 is a set of conditional independences on the considered variables. These are used to orient some edges of the graph according to the d-separation rule and the acyclic properties of DAGs. The result is a PDAG, i.e., a graph that consists of both directed and undirected edges, in which edges that could not be oriented with the previous procedure, are left undirected. The PDAG is then converted into a DAG by an algorithm proposed by Dor and Tarsi (1992). In detail, while keeping the v- structures, the algorithm starts by searching for a sink node in the obtained PDAG, G' say. A vertex s is called a sink if it satisfies the following properties:

(i) there are no directed edges in G' outward from s;

(ii) for every vertex t adjacent to s, with (s,t) undirected, t is adjacent to all the other vertices which are adjacent to s.

If such vertex exists, all the edges which are incident to s in G' are directed toward s. Then, s and all edges incident to it are removed. This procedure is repeated until all edges have been oriented or such s is not found. The pseudo code of this procedure is given in Algorithm 6.

Algorithm 6 Dor and Tarsi algorithm.

```
Input a PDAG $G$.
Let $G' = G$; $A = G$
while $A$ is not empty do
    repeat
        Select a sink $s$ on $A$,
        if $s$ is not found, stop the algorithm
        if $s$ is found, put direction toward $s$ for all edges connected to $s$ in $G'$;
            remove $s$ and all the edges incident to $s$ from $A$.
until $A$ is empty.
return $G'$.
```

The second strategy takes as input the sets of neighbours $N(j),\ j = 1, \ldots, p$, returned from the PNS step, and employs a greedy estimation procedure to estimate the sets of potential parents. We start with an empty DAG, and add at each iteration the edge $k \to j$ between nodes k and j (where $k \in N(j)$) corresponding to the largest gain in log-likelihood (or BIC score). More precisely, a score matrix is constructed to keep track of the change in the score function, where element in position (k,j) indicates how much the log-likelihood (or BIC) increases after adding the edge $k \to j$. For simplicity, here we just consider the representation of the log-likelihood score. Algorithms using the BIC score can be written similarly.

$$\texttt{scoremat}[k,j] = \begin{cases} \ell_j(\hat{\theta}_{pa(j)\cup\{k\}}, \mathbf{x}_{pa(j)\cup\{k\}}) - \ell_j(\hat{\theta}_{pa(j)}, \mathbf{x}_{pa(j)}) & \text{if} \quad k \in N(j) \\ -\texttt{Inf} & \text{if} \quad k \notin N(j). \end{cases}$$

At each iteration, an edge is added corresponding to the largest value in the score matrix. For example, Figure 5.2 left is an example of a score matrix showing the change in the log-likelihood score for the graph in the right panel of Figure 5.1, in which the largest gain (0.8) corresponds to addition of an oriented edge from node 3 to node 4 (Figure 5.2 right).

-	-	-	-	-
-	-	-	0.4	0.7
-	-	-	0.8	0.3
-	0.1	0.5	-	0.6
-	0.3	0.5	0.2	-

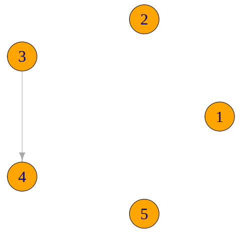


FIGURE 5.2: An example of adding edge 3 → 4 based on calculating the score matrix.

After the addition of an edge, only the j-th column of the score matrix needs to be updated, since the log-likelihood is decomposable as the sum of partial log-likelihoods over all nodes. In order to avoid cycles, we remove from the matrix all values corresponding to paths and inverse paths on the current graph. The maximum number of iterations is $p(p-1)/2$ corresponding to the iterations needed to achieve a fully connected DAG. The potential parent set of each node is defined to be the parent set on the resulting DAG. The pseudo code of this procedure is given in Algorithm 7.

Algorithm 7 orienting edges based on score matrix algorithm.

```
Input sets of neighbours $N(j),\ j = 1, \dots, p$.
Let $G'$ be an empty DAG on $p$ nodes.
Calculate score matrix scoremat.
while $sum$(scoremat != −Inf) > 0 do
    repeat
        Select the maximum value scoremat[$i, j$] in scoremat.
        Add the edge $i \to j$ to $G'$.
        Replace values corresponding to paths and inverse paths on $G'$ by −Inf;
        Update $j$-th column of the score matrix..
until $sum$(scoremat != −Inf) = 0.
return the potential parent sets obtained from $G'$.
```

Such greedy search is prone to mistakes, especially when the sample size is small. To improve the accuracy of Step 2, we propose to employ a bootstrapping technique. In detail, B samples of size m, $S_1, \dots, S_B$ are generated from $\mathbf{x}_1, \dots, \mathbf{x}_n$. For each subsample $S_i,\ i = 1, \dots, B$, the potential parent sets are estimated by applying the above described procedure. The final parent set of each node is defined as the set of nodes most frequently selected in the boostrap replications. In our experiment, we selected nodes appearing at least 60% of the times.

Step 2 could be made more efficient by adding an upper limit on the number of parents that each node can have. Such value could be identified by some prior knowledges on data; in absence of any prior knowledge, it could be set equal to the number of connected nodes after performing PNS.

Clearly, if the PNS step is not implemented, the set of neighbours of node j is $N(j) = V \backslash \{j\},\ j = 1, \dots, p$. The above described procedure can still be performed to find potential parent sets, but the whole procedure is more expensive in terms of computational time.

5.1.3 Step 3: pruning of the DAG

The estimate resulting from (Step 1 and) Step 2 could be a super DAG of the true DAG, i.e., the true DAG is a subgraph of the estimate. As a consequence, edges identified through Step 1 and Step 2 might contain some additional edges. The purpose of this third step is to try to further refine the structure.

We consider two strategies for pruning DAGs: (i) sparse regression techniques; (ii) significance testing procedures. For the first strategy, a number of methods have been proposed in Friedman *et al.* (2010). We apply l_1- penalized regressions at each node on the set representing the potential parent set $\widehat{pa}(s),\ s = 1, \dots p$, estimated in Step 2. In detail, consider a l_1- regularized conditional log-likelihood, i.e.,

$$\hat{\boldsymbol{\theta}}_{\widehat{pa}(s)} = \text{argmin}_{\boldsymbol{\theta}_{\widehat{pa}(s)} \in \mathbb{R}^{|\widehat{pa}(s)|-1}} l(\boldsymbol{\theta}_{\widehat{pa}(s)}, \mathbb{X}_s; \mathbb{X}_{\widehat{pa}(s)}) - \lambda \|\boldsymbol{\theta}_{\widehat{pa}(s)}\|_1.$$

Given the solution $\hat{\boldsymbol{\theta}}_{\widehat{pa}(s)}$, the set of parents of node s is given by

$$\widetilde{pa}(s) = \{t \in \widehat{pa}(s) : \quad \hat{\theta}_{st} \neq 0\}.$$

For the second strategy, we employ significant tests, for example Wald type tests on the parameters $\theta_{st},\ t \in \widehat{pa}(s)$ (see Assumption (3.2)). We test, at some pre-specified significance level, the null hypothesis $H_0 : \theta_{st|\widehat{pa}(s)} = 0$ (see Section 3.1). If the null hypothesis is not rejected, the edge $t \rightarrow s$ is considered to be absent from the graph.

5.2 Empirical study

As in previous chapters, here we study the performance of learnDAG on simulated data, and we compare it with other algorithms. We note that learnDAG allows a great degree of flexibility by accommodating different choices in the various steps. Here, we consider the version of learnDAG based on the the log-likelihood score, since our simulation study showed that results for learnDAG using BIC score look similar. The three possible combinations considered are specified in the following list.

For data generation, we followed the same simulation plan presented in Section 4.4. Moreover, we applied the same data transformations as in Section 4.4 to make the counts compatible with algorithms built for non Poisson data. Competing methods are evaluated w.r.t. their ability to reconstruct the true graph underlying the data.

We considered the following algorithms, listed along with specifications, if needed, of tuning parameters. In this study, we also specified an additional input, i.e., the upper limit for the set of parents, m, which was set equal to $m = 5$ for $p = 10$ and to $m = 3$ for $p = 100$.

- **llearnDAG**: learnDAG using log-likelihood score, LPGM algorithm and k-fold cross validation ($k = 10$) in Step 1, $B = 250$;
- **plearnDAG**: learnDAG using log-likelihood score, PC-LPGM algorithm in Step 1, $B = 250$;
- **olearnDAG**: learnDAG using PC-LPGM algorithm in Step 1, and Algorithm 7;
- **ODS**: Overdispersion score algorithm using k-fold cross validation ($k = 10$);
- **PDN**;
- **PClog**: PC algorithm applied to log transformed data, using Gaussian conditional independent tests at the 1% significance level, and Algorithm 7;
- **MMHC**: Max Min Hill Climbing algorithm applied to data categorized by mixture models, using χ^2 tests of independence at the 1% significance level.

It is worth to note that for graphs of size $p = 100$, the LPGM algorithm [Allen and Liu (2013)] using the stability selection criterion can produce extremely sparse DAGs (see Table 3.2). Consequently, "true parents" are not included in parent sets after performing PNS. A solution to this problem could be to employ lasso or elastic-net regularization path regressions [Friedman *et al.* (2009)]. Here, we employed lasso regressions and a 10-fold cross validation to choose the regularization parameter λ in place of StARS.

Results

For the two considered vertex cardinalities, i.e., $p = 10, 100$, and for the chosen sample sizes, i.e., $n = 200, 1000, 2000$, Table 5.1 and Table 5.2 report, respectively, Monte Carlo means of TP, FP, FN, PPV and Se for each of considered method. Each value is computed as an average of the 1500 values obtained by simulating 500 samples for each of the three networks. Results disaggregated by network type are given in Appendix B, Tables B.7, and Tables B.8. A graphical representation of the results is offered in Figures 5.3, 5.4, 5.5 and 5.6. These results show that our proposed algorithm is competitive with the other approaches in terms of reconstructing the structure.

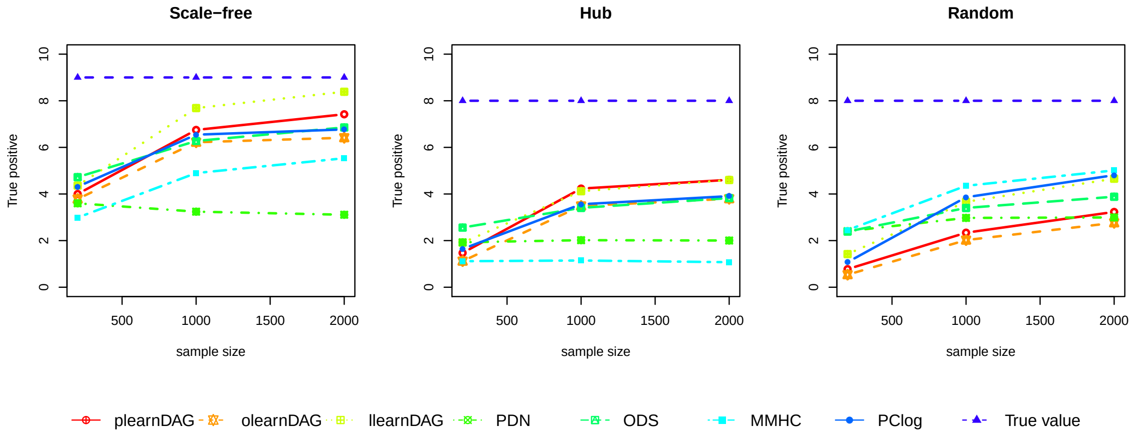


FIGURE 5.3: Number of TP edges recovered by plearnDAG; olearnDAG; llearnDAG; PDN; ODS; MMHC; PClog for networks in Figure 4.1 ($p = 10$) and sample sizes $n = 200$, 1000, 2000.

As far as learnDAG is concerned, in this simulation setting, the use of the log-likelihood score together with the LPGM approach in the step 1 (see llearnDAG), often gives rise to the highest TP value. Differences among various combinations in learnDAG are more evident when $p = 100$. One possible explanation is that the LPGM approach outperforms the PC-LPGM approach in restricting the candidate parent sets when the number of observations is small compared to the number of observations needed to get convergence (see Section 3.5). Besides, olearnDAG using Dor and Tarsi algorithm (Algorithm 7) often has higher standard deviation (see Appendix B, Tables B.7, and Tables B.8). This result is not surprising since the use of the log-likelihood score together with a bootstrap procedure, as done by llearnDAG and plearnDAG, gives rise to more stable results.

As expected, known Poisson algorithms, i.e, ODS, and PDN, reach less accurate results in these scenarios, and their performance is coherent with results stated in previous

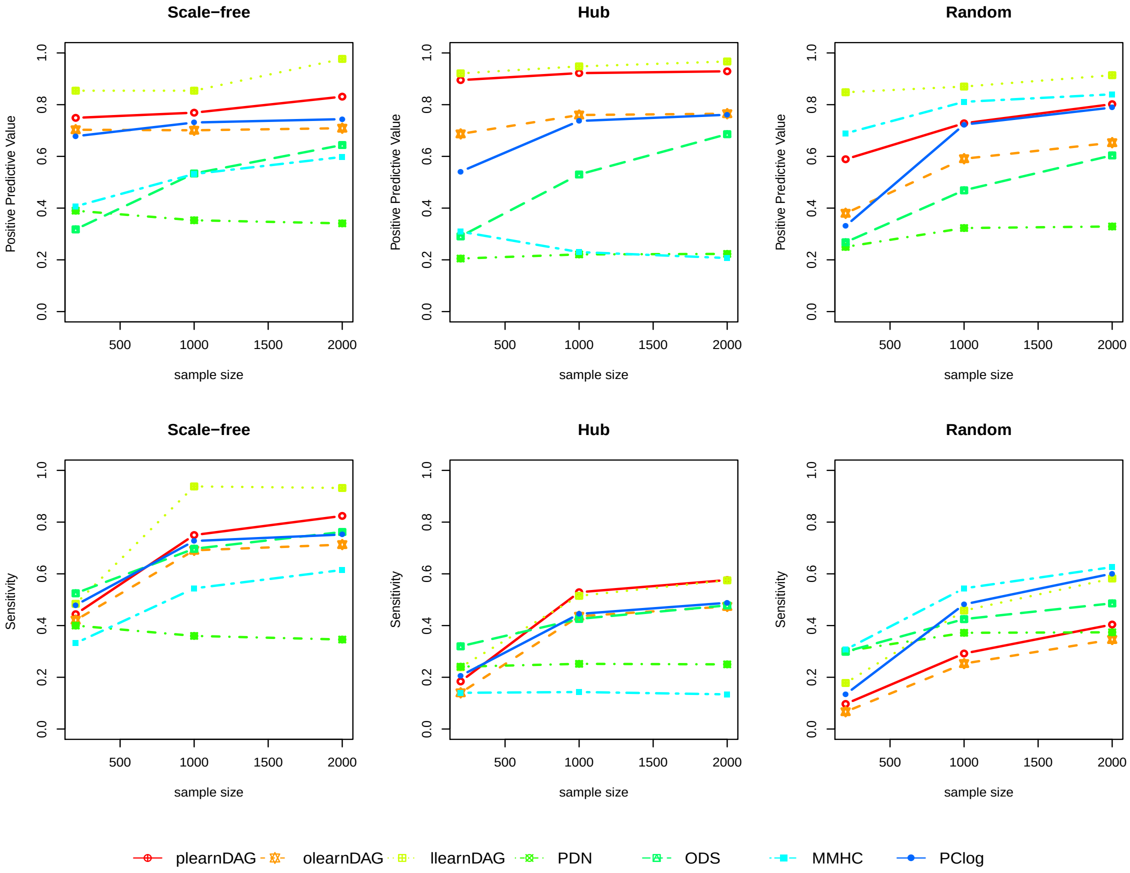


FIGURE 5.4: PPV (first panel row) and Se (second panel row) for plearnDAG; olearnDAG; llearnDAG; PDN; ODS; MMHC; PClog for networks in Figure 4.1 ($p = 10$), sample sizes $n = 200$, 1000, 2000.

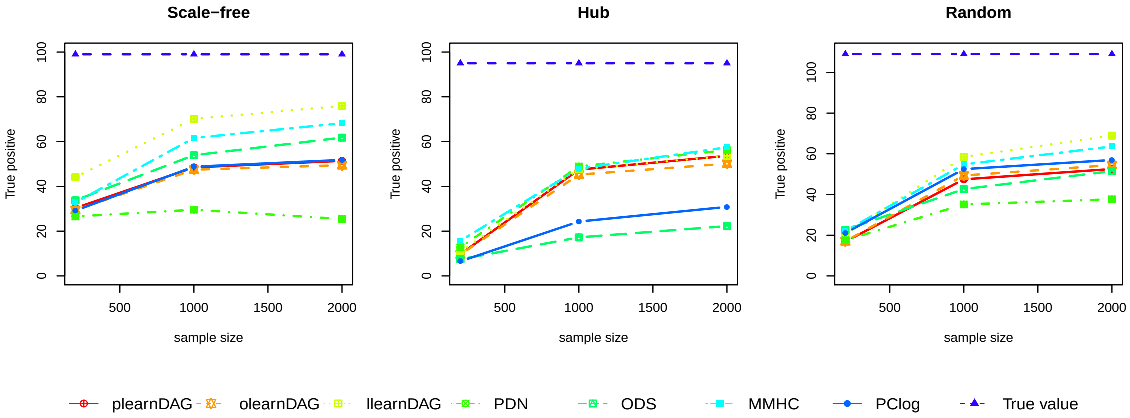


FIGURE 5.5: Number of TP edges recovered by plearnDAG; olearnDAG; llearnDAG; PDN; ODS; MMHC; PClog for networks in Figure 4.2 ($p = 100$) and sample sizes $n = 200$, 1000, 2000.

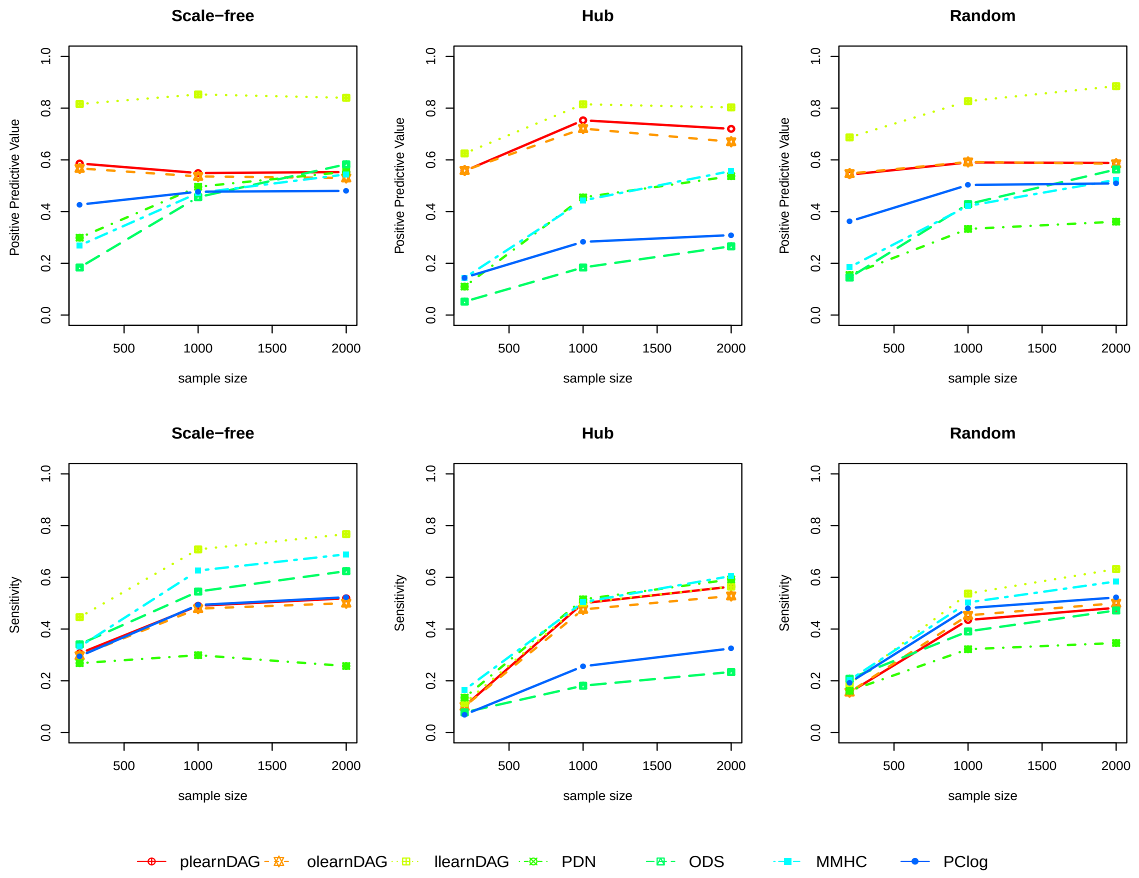


FIGURE 5.6: PPV (first panel row) and Se (second panel row) for plearnDAG; olearnDAG; llearnDAG; PDN; ODS; MMHC; PClog for networks in Figure 4.2 ($p = 100$), sample sizes $n = 200$, 1000, 2000.

Chapter.

Finally, the non Poisson based algorithms, i.e., MMHC, and PClog, perform reasonably well with an inferior score with respect to the leading algorithms.

TABLE 5.1: Monte Carlo marginal means of TP, FP, FN, PPV, Se obtained by simulating 500 samples from each of the three networks shown in Figure 4.1 ($p = 10$).

λ_{noise}	n	Algorithm	TP	FP	FN	PPV	Se
	200	plearnDAG	2.083	0.684	6.251	0.747	0.242
		olearnDAG	1.815	0.975	6.519	0.596	0.209
		llearnDAG	2.565	0.418	5.769	0.875	0.300
		PDN	2.645	6.805	5.689	0.282	0.314
		ODS	3.225	8.285	5.109	0.293	0.381
		MMHC	2.186	2.822	6.147	0.468	0.259
		PClog	2.345	1.703	5.989	0.517	0.273
	1000	plearnDAG	4.440	1.081	3.893	0.806	0.524
		olearnDAG	3.910	1.673	4.423	0.684	0.460

Table 5.1 – continued from previous page

λ_{noise}	n	Algorithm	TP	FP	FN	PPV	Se
		llearnDAG	5.157	0.440	3.177	0.919	0.609
		PDN	2.745	6.449	5.589	0.299	0.328
		ODS	4.361	4.386	3.973	0.511	0.516
		MMHC	3.465	3.089	4.868	0.524	0.410
		PClog	4.655	1.711	3.679	0.730	0.552
	2000	plearnDAG	5.086	0.869	3.247	0.854	0.601
		olearnDAG	4.325	1.702	4.009	0.709	0.511
		llearnDAG	5.881	0.275	2.453	0.953	0.696
		PDN	2.703	6.397	5.631	0.298	0.323
		ODS	4.855	2.830	3.479	0.644	0.575
		MMHC	3.873	2.962	4.460	0.548	0.459
		PClog	5.162	1.619	3.171	0.767	0.614

Table 5.2: Monte Carlo marginal means of TP, FP, FN, PPV, Se obtained by simulating 500 samples from each of the three networks shown in Figure 4.2 ($p = 100$).

λ_{noise}	n	Algorithm	TP	FP	FN	PPV	Se
	200	plearnDAG	21.027	15.806	79.609	0.566	0.209
		olearnDAG	20.633	16.223	80.004	0.559	0.205
		llearnDAG	28.593	8.939	72.044	0.729	0.285
		PDN	18.969	90.596	82.031	0.188	0.188
		ODS	21.289	142.525	79.711	0.127	0.209
		MMHC	23.533	93.470	77.467	0.199	0.233
		PClog	18.936	38.365	82.064	0.311	0.186
	1000	plearnDAG	47.812	29.507	53.188	0.631	0.475
		olearnDAG	47.347	30.902	53.653	0.616	0.469
		llearnDAG	58.777	11.786	42.223	0.832	0.582
		PDN	37.865	56.137	63.135	0.428	0.379
		ODS	37.931	66.563	63.069	0.356	0.372
		MMHC	54.885	68.566	46.115	0.446	0.544
		PClog	41.847	55.726	59.153	0.421	0.410
	2000	plearnDAG	52.307	34.677	48.329	0.608	0.521
		olearnDAG	51.012	37.322	49.625	0.583	0.507
		llearnDAG	67.935	12.652	32.701	0.842	0.675
		PDN	39.765	47.931	61.235	0.484	0.398
		ODS	45.166	49.116	55.834	0.471	0.443
		MMHC	63.094	53.902	37.906	0.541	0.626
		PClog	46.531	60.152	54.469	0.433	0.457

5.3 Discussion

We have proposed an unguided structure learning algorithm and compared it to a number of different approaches. A key strategy of our approach is to decouple potential parent sets from edge selection in DAGs. Once the potential parent sets are estimated, learning DAGs reduces to a simple set of p regression problems. The use of the PC-LPGM algorithm in Step 1, as well as significance tests in Step 3, makes our algorithm more efficient than the latest Poisson structure learning algorithm, i.e., ODS. Furthermore, we note that the restriction to the parent sets in addition to a boosting approach makes the proposed algorithm more stable, with low standard deviation (see Appendix B, Tables B.7, and Tables B.8).

It is worth to note that final performance of learnDAG depends on the chosen combination. For example, we have noticed that using PC-LPGM in Step 1 is better than using LPGM when the true graph is not too sparse. Indeed, we simulated a random graph for $p = 100$, and an edge probability (0.03) leading to a better performance of plearnDAG and of olearnDAG over llearnDAG (see Table 5.3).

As stated before, we can dismiss Step 1 with small graphs. Sometimes, experiment results show that the distance between the DAG recovered without Step 1 and the true DAG is smaller than the one obtained by implementing Step 1, since PNS can produce false negatives. However, the PNS step is essential for the algorithm to be feasible when p is large, as the computational cost is significantly reduced.

The log-likelihood score is used as the key to orient directions of DAGs. This makes our algorithm quite flexible in applications with respect to the overdispersion scoring in Park and Raskutti (2015), which fails with data coming from distributions which are not Poisson. Moreover, we can adapt the proposed method to various likelihood-based scores (like BIC or AIC).

Finally, Theorem 7 in Meek (1995) shows that violations of faithfulness in discrete distributions are Lebesgue measure zero. In other words, almost all discrete distributions that are Markov to a given graph are faithful to it. If we restrict our attention to the subspace on which the faithfulness condition is satisfied, the consistency of the proposed algorithm can be deduced from statistical theory. In detail, for the first combination of learnDAG, i.e., olearnDAG, under the same assumptions in Chapter 3 and Faithfulness condition, the proof of Theorem 4.3.1 also covers the issue of estimating the correct skeleton and separation sets. Then, the estimated PDAG from Step 1 convergences to the true DAG (thanks to Identifiability property in Chapter 2). For other combinations, i.e., plearnDAG, llearnDAG, the greedy search in Step 2 guarantees that the algorithm identifies the true structure up to an equivalent class as the number of observations goes to infinity when a consistent scoring criterion is used [Chickering and Meek (2002)]. As BIC is consistent scoring criterion [Haughton *et al.* (1988)], the estimated graph convergences to the true DAG since the Poisson model is identifiable (see Section 2.2).

Table 5.3: Monte Carlo marginal means of TP, FP, FN, PPV, Se obtained by simulating 500 samples from random network with edge probability 0.03, $p = 100$.

λ_{noise}	n	Algorithm	TP	FP	FN	PPV	Se
	200	plearnDAG	83.880	7.973	166.120	0.913	0.336
		olearnDAG	83.117	8.600	166.883	0.905	0.332
		llearnDAG	75.947	16.410	174.053	0.823	0.304
		PDN	34.520	73.837	215.480	0.321	0.138
		ODS	73.630	255.840	176.370	0.225	0.295
		MMHC	70.267	81.683	179.733	0.463	0.281
		PClog	76.877	23.900	173.123	0.763	0.308
	1000	plearnDAG	208.497	5.673	41.503	0.974	0.834
		olearnDAG	212.967	7.080	37.033	0.968	0.852
		llearnDAG	182.130	18.250	67.870	0.909	0.729
		PDN	42.820	48.152	207.180	0.467	0.171
		ODS	130.940	181.781	119.060	0.419	0.524
		MMHC	186.703	65.427	63.297	0.741	0.747
		PClog	208.792	13.590	41.208	0.939	0.835
	2000	plearnDAG	223.300	5.850	26.700	0.974	0.893
		olearnDAG	232.493	7.397	17.507	0.969	0.930
		llearnDAG	198.037	19.530	51.963	0.910	0.792
		PDN	41.504	43.589	208.496	0.479	0.166
		ODS	152.733	143.198	97.267	0.517	0.611
		MMHC	217.190	50.053	32.810	0.813	0.869
		PClog	233.426	12.167	16.574	0.951	0.934

Chapter 6

Conclusions

The problem of learning the structure of a DAG from a set of given count data is an NP hard problem, as the number of possible structures grows exponentially in the number of nodes. Here, we tackled such problem when count data are available. Our first results, presented in this thesis, show promise. Many open problems await future research, and we mention some of them here.

Firstly, the proposed algorithms could be extended to other count distributions, as, for example the negative binomial distribution. The same algorithms could be customized making use of the negative binomial assumption instead of the Poisson one whenever needed. It should be possible and would be interesting to obtain statistical results to guarantee consistency of the estimators in this wider setting.

Secondly, more work needs to be done to extend our algorithms to general models, for example considering $f_s(.)$ an arbitrary function. We might for example consider Poisson generalized additive models, where the conditional mean can be written as:

$$f_j(\mathbf{x}_{V\setminus j}) = \sum_{k\neq j} f_{jk}(x_k) + \epsilon_j,$$

where $f_{jk}(.) \neq 0$ if and only if there is a directed edge $k \rightarrow j$ in G, and $\epsilon_1, \ldots, \epsilon_p$ are independent with $\epsilon_j \sim N(0, \sigma_j^2), \quad \sigma_j^2 > 0$.

Thirdly, in this thesis, we adopted a local approach to model specification leading to possible non existence of the joint distribution when undirected graphs are considered that contain both positive and negative dependencies. It should be interesting to study consistency of our class of structure learning algorithms for models that guarantee the existence of the unique joint distribution, as, for example, TPGM, SPGM, QPGM [Yang *et al.* (2013)]. Yang *et al.* (2015) proved that any exponential distribution produces Assumption 3.2.4, Assumption 3.2.5. Hence, we should investigate whether these results remain in this particular modifications of the Poisson family.

The proposed unguided structure learning algorithm for Poisson DAGs, i.e., learnDAG contains three steps, and each of them could be customized. Hence, another interesting avenue for future work is to exploit alternative procedures at each step. For example, in Step 1, we could consider testing the neighbourhoods as in Verzelen and Villers (2009).

In this thesis, we do not discuss run-time complexity of our algorithms. Clearly, further work should be developed to explore this issue.

Appendix A

Here, we briefly review some popular non-Poisson structure learning algorithms, that were used in this thesis.

A.1 K2 algorithm

The K2 algorithm, a representative of the scoring-based approach, was introduced by Cooper and Herskovits (1992), and is known as one of the first efficient solutions to the problem of learning DAGs from data. Before its definition, DAGs were usually constructed by hand, in close collaboration with domain experts. Therefore, a number of limitations raised as the consequence of this approach, such as the size of the considered network, the availability of the domain knowledge. These drawbacks feed the need of developing a procedure that would construct high dimensional DAGs automatically. Cooper and Herskovits (1992) defined a new Bayesian score function, the K2 score, that scores individual DAGs reflecting how well they fit the observed data. Then, structure learning problem boils down to finding the structure that maximizes the score. However, as a function of the number of nodes, the number of possible structures grows exponentially. Thus, an exhaustive enumeration of all network structures is not feasible in most domains. To overcome this difficulty, the K2 algorithm takes as an input in addition to the data, the ordering of variables. The key idea behind the algorithm is using a Bayesian scoring function, i.e., the K2 score, that reflects how well individual graphs fit the data. More precisely, the analytical expression is given in the following result.

Theorem A.1.1. Let there be n observations of p-discrete variables $X_1, \dots, X_p$, where X_i has r_i possible value assignment: $(v_{i1}, \dots, v_{ir_i})$. Let G be a DAG containing these variables. Let pa_i denote the set of parents of X_i in G. Let ω_{ij} denote the j-th unique realization of pa_i. Suppose there are q_i such unique realizations. Define N_{ijk} to be the number of observations in which X_i assumes the value v_{ik} and pa_i has the value w_{ij}.

Let

$$N_{ij} = \sum_{k=1}^{r_i} N_{ijk}.$$

If the prior distribution is uniform over a set of all possible DAGs on p nodes, then K2 score of the structure G corresponds up to a constant to its posterior probability and is given by

$$g(G) = \prod_{i=1}^{p} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!. \tag{A.1}$$

In summary, Theorem A.1.1 states that under very mild conditions regarding the uniform prior distribution, the posterior probability of a DAG structure is given in the formula (A.1). For the reader interested in the proof, we refer to the original work of Cooper and Herskovits (1992).

The K2 algorithm aims at searching for the graph that maximizes the K2 score among the space of all possible DAGs. However, the search is not trivial, since the number of possible structures grows exponentially with the number of nodes. Thus, to reduce the search space, the K2 algorithm also requires a topological ordering of the variables. Once, the order is specified, then a heuristic search method is used. In particular, the algorithm starts from the first node in the topological ordering. For a fixed node, it begins by making the assumption that the parent set is empty, and consider the candidate parent set consists of all variables that precede the fixed node in the ordering. At each iteration, the K2 score is calculated, then variable that gains most the score is added to the parent set. This procedure stops when no addition of a single parent can increase the score (or we get enough m parents, where m is the maximum number of parents that one node can have), and it moves to the next variable in the ordering. The pseudo-code is given in Algorithm 8.

A.2 PC algorithm

The PC algorithm [Spirtes *et al.* (2000)] is the most popular representative of constrained-based algorithms. It is designed to refine the underlying directed graph from given data, based one d-separation and acyclic properties. Starting from a completely undirected graph, edges whose evidence is not supported by the data are in turn removed. Specifically, this procedure consists of a sequence of conditional independence tests.

- For each pair of variables X_i and X_j, test whether they are independent; if so, remove the edge between them, then update the graph G, and move to the first order conditional independence.

Algorithm 8 K2 Algorithm.

```
Input $n$ observations of $p$ variables $X_1, \dots, X_p$, a topological ordering on nodes $Ord$,
  an upper bound for the number $u$ of parents of a node may have.
for $i = Ord[1]$ to $Ord[p]$ do
    $pa_i \leftarrow \emptyset$
    $P_{old} \leftarrow g(i, pa_i)$
    OKToProceed $\leftarrow$ TRUE
    while OKToProceed and $|pa_i| < u$ do
        let $z$ be the node in $pre(x_i) \backslash pa_i$ that maximizes $g(i, pa_i \cup \{z\})$
        $P_{new} \leftarrow g(i, pa_i \cup \{z\})$
        if $P_{new} > P_{old}$ then
            $P_{old} \leftarrow P_{new}$
            $pa_i \leftarrow pa_i \cup \{z\}$
        else OKToProceed $\leftarrow$ FALSE
        end if
    end while
    print "parents of node" $x_i$, "are" $pa_i$.
end for
```

- For each connected pair X_i and X_j, test $X_i \perp\!\!\!\perp X_j | X_k$, for each $k \in N(i) \backslash \{j\}$. If the hypothesis is not rejected for some k, remove the edges between X_i and X_k and move to the second order conditional independence.
- Continue with higher order conditional independence tests until further conditioning is impossible.

If all the tests are done, a process to orient some edges of the undirected graph can be performed by using simple probabilistic considerations, as well as the property of acyclicity. As a consequence, the output of the PC algorithm is a PDAG, representative of a certain equivalence class.

Results of the PC algorithm, moreover, depend on the choice of conditional independence tests, which in turn, depends on the nature of considered variables. The most popular choice are χ^2 test of independence in case of categorical variables [Spirtes *et al.* (2000)]; and the test on partial correlation under the Gaussian assumption [Kalisch and Bühlmann (2007)]. The sparsity of the final structure depends on the choice of the significance level of the tests: the higher the significance level, the lower the number of edges.

A.3 VSL algorithm

In Gaussian graphical models, the global, local and pairwise Markov property are equivalent. Thus, two major directions of structure learning are covariance selection, and neighbourhood selection. We consider a representative of the latter: Variable selection with the lasso of Meinshausen and Bühlmann (2006). Here, the problem of structure learning is translated into an optimization problem with a residual squared error and a lasso constraint.

With neighbourhood selection approach, learning graphical structures boils down to estimate the conditional independence restrictions separately for each node in the graph, which is equivalent to variable selection for Gaussian linear models. For each node $s \in V$, the neighbourhood $N(s)$ is defined as the variables corresponding to non-zero coefficients in the prediction of X_s on all other random variables $\mathbf{X}_{V\backslash\{s\}}$. Let $\hat{\boldsymbol{\beta}}_s = (\hat{\beta}_{st})_{t \in V\backslash\{s\}}$ be the set of coefficients for optimal prediction,

$$\hat{\boldsymbol{\beta}}_s = \text{argmin}_{\ \boldsymbol{\beta}_s \in \mathbb{R}^{p-1}} \mathbb{E}(X_s - \sum_{t \in V\backslash\{s\}} \beta_{st} X_t)^2. \tag{A.2}$$

Then, the set of neighbours of node $s \in V$ is defined as

$$N(s) = \{t \in V\backslash\{s\} : \quad \hat{\beta}_{st} \neq 0\}.$$

A natural way to estimate the vector $\hat{\boldsymbol{\beta}}_s$ in Equation (A.2) is minimizing the residual squared error, i.e.,

$$\hat{\boldsymbol{\beta}}_s = \text{argmin}_{\ \boldsymbol{\beta}_s \in \mathbb{R}^{p-1}} \left(\frac{1}{n} \sum_{i=1}^{n} (x_{is} - \sum_{t \neq s} \beta_{st} x_{it})^2 \right).$$

Prediction accuracy can sometimes be improved by shrinking or setting to 0 some coefficients. Meinshausen and Bühlmann (2006) proposed neighbourhood selection with the lasso, i.e.,

$$\hat{\boldsymbol{\beta}}_s^{\lambda} = \text{argmin}_{\ \boldsymbol{\beta}_s \in \mathbb{R}^{p-1}} \left(\frac{1}{n} \sum_{i=1}^{n} (x_{is} - \sum_{t \neq s} \beta_{st} x_{it})^2 + \lambda \|\boldsymbol{\beta}_s\|_1 \right).$$

Structure learning is done after performing p standard regressions which can be solved easily by using the algorithm, knonwn as glmnet, proposed by Friedman *et al.* (2009). For each value of the regularization parameter λ, neighbourhood sets are specified as follows

$$\hat{N}^{\lambda}(s) = \{t \in V\backslash\{s\} : \quad \hat{\beta}^{\lambda}_{st} \neq 0 \text{ or } \hat{\beta}^{\lambda}_{ts} \neq 0\}.$$

Thus, the number of non-zero coefficients or the sparsity of the graph estimates depend on the choice of the regularization parameter λ. Two popular choices for choosing a suitable parameter λ are: cross validation as in Friedman *et al.* (2009), and stability selection criterion (StARS) as in Liu *et al.* (2010), which seeks the regularization parameter λ leading to the most stable set of edges. We will take the latter choice for all algorithms in this work whenever we need to choose a value for λ.

A.4 GLASSO algorithm

The VSL algorithm takes a simple approach to estimate sparse undirected graphical models by fitting a lasso model to each variable (an approximation to the exact problem). Friedman *et al.* (2008) propose a new algorithm for the exact problem, called GLASSO by using the blockwise coordinate descent. Here, the interest is not in considering estimation of the precision matrix $\Gamma = \Sigma^{-1}$ directly, but of the covariance matrix Σ instead. Then, one can deduce an estimate of the inverse covariance matrix relatively cheaply.

Recently, a number of approaches to structure learning that focus on the problem of estimating the graph in high dimensional settings under a sparsity assumption. Banerjee *et al.* (2008) proposed an estimator based on regularized maximum likelihood minus an l_1 constraint on the entries of Γ, called a block coordinate descent algorithm. Friedman *et al.* (2008) use this approach as a launching point to develop a new algorithm for estimate Σ. Precisely, let W be the estimate of Σ, we partition W and S as follows

$$W = \begin{pmatrix} W_{11} & w_{12} \\ w_{12}^T & w_{22} \end{pmatrix}, \qquad S = \begin{pmatrix} S_{11} & s_{12} \\ s_{12}^T & s_{22} \end{pmatrix} \tag{A.3}$$

Friedman *et al.* (2008) translated the problem of maximizing the penalized log-likelihood problem to a lasso least square problem

$$\min_{\boldsymbol{\beta}} \left\{ \frac{1}{2} \| W_{11}^{1/2} \boldsymbol{\beta} - b \|^2 + \lambda \|\boldsymbol{\beta}\|_1 \right\}, \tag{A.4}$$

where $b = W_{11}^{-1/2} s_{12}$, which can be solved by the standard lasso algorithm. Thus, GLASSO is really simple algorithm. Each time, we estimate a single row and column of Γ. Permuting the rows and columns so that the target column is always the last. This procedure repeat until convergence.

Till now, we just discuss how to estimate the covariance matrix Σ. To recover the structure of the underlying graph, we need the estimate of precision matrix Γ. However, this matrix can be easily deduced by expending the relation between Γ and the matrix estimate W, i.e., $W\Gamma = I$. Precise expression of the estimate matrix $\hat{\Gamma}$ can be found in Friedman *et al.* (2008).

A.5 Extension to the nonparanormal model

Exact normality assumption restricts the application of Gaussian models. Liu *et al.* (2009) relax this assumption by transforming the variables of interest through some smooth functions. This transformation extends the class of Gaussian distributions to a strictly larger class, called the nonparanormal class, which preserves the conditional independence relations. Thus, structure learning of the original variables can be inferred from the structure of the transforming data.

Definition A.5.1. A random vector $\mathbf{X} = (X_1, \dots, X_p)$ has a nonparanormal distribution if there exists a function $f = (f_1, \dots, f_p)$ s.t. $Z = f(\mathbf{X}) \sim N(\mu, \Sigma)$, where $f(\mathbf{X}) = (f_1(X_1), \dots, f_p(X_p))$. We then write

$$\mathbf{X} \sim NPN(\mu, \Sigma, f).$$

Liu *et al.* (2009) proved that when functions $f_s,\ s \in V$ are monotonic and differentiable functions, the nonparanormal family is equivalent to the Gaussian copula family. Then, the conditional independence relations are still encoded by zero entries of $\Gamma = \Sigma^{-1}$, i.e.,

$$X_s \perp\!\!\!\perp X_t | \mathbf{x}_{V\backslash\{s,t\}} \Leftrightarrow \gamma_{st} = 0.$$

Therefore, learning the graphical structure that encodes the conditional independence relationships between the random variables $X_1, \dots, X_p$ now is translated into learning Gaussian graphical model related to multivariate normal distributions $f(\mathbf{X}) = (f_1(X_1), \dots, f_p(X_p))$, which can be solved by GLASSO.

The problem here is estimating the smooth function $f(.)$. Liu *et al.* (2009) give an estimator for the transformation function using a Winsorized estimator of the marginal distribution function $F_s, s \in V$. Let $\hat{F}_s$ be the empirical marginal distribution of F_s.

Then, the Winsorized estimator of F_s is defined as follows,

$$\tilde{F}_s(x) = \begin{cases} \delta_n & \text{if} \quad \hat{F}_s(x) \leq \delta_n \\ \hat{F}_s(x) & \text{if} \quad \delta \leq \hat{F}_s(x) \leq 1 - \delta_n \\ (1 - \delta_n) & \text{if} \quad \hat{F}_s(x) > 1 - \delta_n. \end{cases} \tag{A.5}$$

with $\delta_n = \dfrac{1}{4n^{1/4}\sqrt{\pi \log n}}$ as the truncated value. Then the transformation function f_s is

$$\tilde{f}_s(x) = \hat{\mu_s} + \hat{\sigma_s}\Phi^{-1}(\tilde{F}_s(x)),$$

where $\hat{\mu}_s, \hat{\sigma}_s$ are the sample mean and sample variance of random variable X_s respectively.

In summary, the nonparanormal algorithm, called NPN-Copula has two step procedures:

1. For each variable, replace the observations by their respective transformed data.
2. Apply GLASSO algorithm on the transformed data to estimate the undirected graph.

Liu *et al.* (2012) also propose an algorithm to learn graphical structure underlying the nonparanormal assumption, called NPN- skeptic. The name of the method stands for Spearman/Kedall estimates preempt transformations to infer correlation. It is named after the main idea behind, i.e., exploiting Spearman's ρ and Kendall's τ statistics to directly estimate the unknown correlation matrix, without explicitly calculating the marginal transformations f_s.

The NPN- SKEPTIC algorithm is formed by keeping the second step in the NPN-Copula algorithm while replacing Step 1 by an alternative procedure. The key of this replacing is the connection between Spearman's ρ and Kendall's τ to the underlying Pearson correlation coefficient Σ^0 in Kruskal (1958). Assuming $X \sim NPN(\mu, \Sigma^0, f)$, then

$$\Sigma^0_{st} = 2\sin\left(\frac{\pi}{6}\rho_{st}\right) = \sin\left(\frac{\pi}{2}\tau_{st}\right). \tag{A.6}$$

Hence, by replacing the empirical estimations $\hat{\rho_{st}}$ (or $\hat{\tau_{st}}$) of Spearman's ρ_{st} (or Kendall's τ_{st}) to Equation (A.6), we get an estimate of the correlation matrix Σ^0_{st}. This estimate is then plugged into GLASSO or alternative algorithms such as CLIME (see Cai *et al.* (2011)), the graphical Dantzig selector (see Yuan (2010)) to obtain the final estimate of the inverse correlation matrix and graph. In this work, we just consider NPN-skeptic with GLASSO, so when we say NPN-skeptic, it means that we use GLASSO in computing the estimate of the precision matrix.

A.6 MMHC algorithm

The MMHC algorithm, introduced by Tsamardinos *et al.* (2006), is used to learn structure DAG. This algorithm combines ideas from local learning, constraint based, and search-and-score techniques. They first learn the undirected graph underlying data by using a local search algorithm called Max-Min Parents and Children (MMPC). In particular, fixed a target variable X_i, MMPC identifies all the edges that connect to X_i as follows:

- In the forward phase, variables enter sequentially into the candidate set of parents and children (CPC) of X_i by use of a heuristic function. At each iteration, the variable mostly associated to X_i is selected until no other variables are founded to be associated to the target.

- In the backward phase, for each X_j in the CPC of X_i, the test $X_i \perp\!\!\!\perp X_j | S$ is performed, where $S \subset CPC$. If there exists some S such that the hypothesis is not rejected, remove X_j from CPC.

- Checking symmetry: the aim of this procedure is to remove false positive in CPC, based on checking the parent and children relation.

As every other constraint based learning algorithm, MMPC algorithm depends on the use of conditional independence tests. The χ^2 test independence in contingency tables is also employed here, since it is relatively easy to compute. However, it is only asymptotically correct for a general discrete multinomial distribution.

The output of MMPC algorithm, the skeleton of a Bayesian network, will be the input of the oriented procedure in the next step. Here, search-and-score methods search over a space of possible structures, guided by a scoring function. Specifically, a greedy hill-climbing search in the space of Bayesian networks begins with an empty graph. Edges are added, deleted, or reversed direction such that the BDeu score [Heckerman *et al.* (1995)] increases largest. The BDeu score could be seen as the posterior probability of the structure learned under some certain conditions. Finally, we note, MMHC however just searches only on the set of edges that was discovered by MMPC.

Appendix B

B.1 Appendix B.1

Table B.1 to Table B.4 report TP, FP, FN, PPV and Se for each of methods considered in Section 3.3. Two different graph dimensions, i.e., $p = 10, 100$, and three graph structures (see Figure 3.1 and Figure 3.2) are considered at one low ($\lambda_{noise} = 5$) and one high ($\lambda_{noise} = 0.5$) SNR levels.

Table B.1: Simulation results from 500 replicates of the undirected graphs shown in Figure 3.1 for $p = 10$ variables with Poisson node conditional distribution and level of noise $\lambda_{noise} = 0.5$. Monte Carlo means (standard deviations) are shown for TP, FP, FN, PPV and Se.

Graph	n	Algorithm	TP	FP	FN	PPV	Se
Scale-free	200	PC-LPGM	6.838 (1.152)	0.048 (0.230)	2.163 (1.152)	0.994 (0.208)	0.760 (0.169)
		LPGM	4.732 (1.407)	0.384 (0.644)	4.268 (1.407)	0.941 (0.097)	0.526 (0.156)
		PDN	5.872 (0.741)	0.182 (0.430)	3.128 (0.741)	0.972 (0.065)	0.652 (0.082)
		VSL	4.625 (2.056)	0.034 (0.181)	4.375 (2.056)	0.996 (0.021)	0.514 (0.228)
		GLASSO	4.502 (1.961)	0.023 (0.151)	4.498 (1.961)	0.997 (0.018)	0.500 (0.218)
		NPN-Copula	5.073 (2.169)	0.034 (0.191)	3.927 (2.169)	0.996 (0.023)	0.564 (0.241)
		NPN-Skeptic	5.030 (2.177)	0.039 (0.230)	3.970 (2.177)	0.994 (0.023)	0.559 (0.242)
	1000	PC-LPGM	9.000 (0.000)	0.071 (0.258)	0.000 (0.000)	0.993 (0.026)	1.000 (0.000)
		LPGM	5.780 (1.253)	0.692 (2.730)	3.220 (1.253)	0.964 (0.135)	0.642 (0.139)
		PDN	5.780 (0.661)	0.000 (0.000)	3.220 (0.661)	1.000 (0.000)	0.642 (0.073)
		VSL	4.954 (2.246)	0.000 (0.000)	4.046 (2.246)	1.000 (0.000)	0.550 (0.250)
		GLASSO	4.889 (2.234)	0.000 (0.000)	4.111 (2.234)	1.000 (0.000)	0.543 (0.248)
		NPN-Copula	5.377 (2.451)	0.000 (0.000)	3.623 (2.451)	1.000 (0.000)	0.597 (0.272)
		NPN-Skeptic	5.232 (2.609)	0.000 (0.000)	3.768 (2.069)	1.000 (0.000)	0.581 (0.290)
	2000	PC-LPGMC	9.000 (0.000)	0.071 (0.278)	0.000 (0.000)	0.993 (0.027)	1.000 (0.000)
		LPGM	7.660 (1.611)	5.180 (4.482)	1.340 (1.611)	0.703 (0.238)	0.851 (0.179)
		PDN	5.658 (0.581)	0.000 (0.000)	3.342 (0.581)	1.000 (0.000)	0.629 (0.065)
		VSL	5.566 (2.381)	0.000 (0.000)	3.434 (2.381)	1.000 (0.000)	0.618 (0.265)
		GLASSO	5.573 (2.381)	0.000 (0.000)	3.427 (2.381)	1.000 (0.000)	0.619 (0.265)
		NPN-Copula	6.055 (2.509)	0.000 (0.000)	2.945 (2.509)	1.000 (0.000)	0.673 (0.279)
		NPN-Skeptic	5.945 (2.710)	0.000 (0.000)	3.055 (2.710)	1.000 (0.000)	0.661 (0.301)
	200	PC-LPGM	6.618 (1.042)	0.104 (0.132)	1.382 (1.042)	0.986 (0.042)	0.827 (0.130)

Table B.1 – continued from previous page

Graph	n	Algorithm	TP	FP	FN	PPV	Se
Hub		LPGM	3.072 (1.124)	0.136 (0.505)	4.928 (1.124)	0.975 (0.077)	0.384 (0.144)
		PDN	6.680 (0.700)	0.560 (0.769)	1.320 (0.700)	0.926 (0.099)	0.835 (0.088)
		VSL	4.316 (1.933)	0.030 (0.171)	3.684 (1.933)	0.995 (0.033)	0.540 (0.242)
		GLASSO	4.212 (1.903)	0.028 (0.177)	3.788 (1.903)	0.995 (0.033)	0.527 (0.238)
		NPN-Copula	4.636 (1.936)	0.024 (0.166)	3.364 (1.936)	0.996 (0.026)	0.580 (0.242)
		NPN-Skeptic	4.506 (2.009)	0.032 (0.187)	3.494 (2.009)	0.995 (0.028)	0.563 (0.251)
	1000	PC-LPGM	8.000 (0.000)	0.122 (0.345)	0.000 (0.000)	0.987 (0.038)	1.000 (0.000)
		LPGM	4.392 (2.669)	1.452 (2.201)	3.608 (2.669)	0.885 (0.169)	0.549 (0.334)
		PDN	7.128 (0.395)	0.000 (0.000)	0.872 (0.395)	1.000 (0.000)	0.891 (0.049)
		VSL	5.908 (1.920)	0.000 (0.000)	2.092 (1.920)	1.000 (0.000)	0.739 (0.240)
		GLASSO	5.842 (1.907)	0.000 (0.000)	2.158 (1.907)	1.000 (0.000)	0.730 (0.238)
		NPN-Copula	6.000 (2.094)	0.000 (0.000)	2.000 (2.094)	1.000 (0.000)	0.750 (0.262)
		NPN-Skeptic	5.818 (2.337)	0.000 (0.000)	2.182 (2.337)	1.000 (0.000)	0.727 (0.292)
	2000	PC-LPGM	8.000 (0.000)	0.132 (0.373)	0.000 (0.000)	0.986 (0.040)	1.000 (0.000)
		LPGM	6.252 (2.688)	2.480 (1.904)	1.748 (2.688)	0.790 (0.151)	0.782 (0.336)
		PDN	7.216 (0.488)	0.000 (0.000)	0.784 (0.488)	1.000 (0.000)	0.902 (0.061)
		VSL	7.110 (1.680)	0.000 (0.000)	0.890 (1.680)	1.000 (0.000)	0.889 (0.210)
		GLASSO	7.068 (1.681)	0.000 (0.000)	0.932 (1.681)	1.000 (0.000)	0.884 (0.210)
		NPN-Copula	7.006 (2.030)	0.000 (0.000)	0.994 (2.030)	1.000 (0.000)	0.876 (0.254)
		NPN-Skeptic	6.794 (2.272)	0.000 (0.000)	1.206 (2.272)	1.000 (0.000)	0.849 (0.284)
Random	200	PC-LPGM	5.492 (1.581)	0.052 (0.231)	2.508 (1.581)	0.991 (0.039)	0.687 (0.198)
		LPGM	3.500 (1.120)	0.244 (0.531)	4.500 (1.120)	0.950 (0.107)	0.438 (0.140)
		PDN	4.800 (0.752)	2.362 (0.817)	3.200 (0.752)	0.675 (0.085)	0.600 (0.094)
		VSL	3.510 (1.655)	0.034 (0.202)	4.490 (1.655)	0.993 (0.040)	0.439 (0.207)
		GLASSO	3.464 (1.601)	0.026 (0.171)	4.536 (1.601)	0.995 (0.036)	0.433 (0.200)
		NPN-Copula	3.934 (1.823)	0.028 (0.165)	4.066 (1.823)	0.995 (0.030)	0.492 (0.228)
		NPN-Skeptic	3.826 (1.859)	0.030 (0.182)	4.174 (1.859)	0.995 (0.031)	0.478 (0.232)
	1000	PC-LPGM	8.000 (0.000)	0.078 (0.283)	0.000 (0.000)	0.991 (0.031)	1.000 (0.000)
		LPGM	5.748 (1.989)	3.584 (3.752)	2.252 (1.989)	0.758 (0.244)	0.718 (0.249)
		PDN	5.066 (0.753)	2.164 (0.634)	2.934 (0.753)	0.703 (0.068)	0.633 (0.094)
		VSL	3.190 (1.963)	0.000 (0.000)	4.810 (1.963)	1.000 (0.000)	0.399 (0.245)
		GLASSO	3.110 (1.897)	0.000 (0.000)	4.890 (1.897)	1.000 (0.000)	0.389 (0.237)
		NPN-Copula	3.434 (2.257)	0.000 (0.000)	4.566 (2.257)	1.000 (0.000)	0.429 (0.282)
		NPN-Skeptic	3.358 (2.351)	0.000 (0.000)	4.642 (2.351)	1.000 (0.000)	0.420 (0.294)
	2000	PC-LPGM	8.000 (0.000)	0.048 (0.214)	0.000 (0.000)	0.995 (0.024)	1.000 (0.000)
		LPGM	7.484 (1.073)	6.256 (2.369)	0.516 (1.073)	0.576 (0.140)	0.936 (0.134)
		PDN	5.068 (0.730)	2.082 (0.716)	2.932 (0.730)	0.713 (0.080)	0.634 (0.091)
		VSL	2.952 (2.011)	0.000 (0.000)	5.048 (2.011)	1.000 (0.000)	0.369 (0.251)
		GLASSO	2.828 (1.886)	0.000 (0.000)	5.172 (1.886)	1.000 (0.000)	0.353 (0.236)
		NPN-Copula	3.356 (2,261)	0.000 (0.000)	4.644 (2.261)	1.000 (0.000)	0.420 (0.283)
		NPN-Skeptic	3.384 (2.321)	0.000 (0.000)	4.616 (2.321)	1.000 (0.000)	0.423 (0.290)

Table B.2: Simulation results from 500 replicates of the undirected graphs shown in Figure 3.1 for $p = 10$ variables with Poisson node conditional distribution and level of noise $\lambda_{noise} = 5$. Monte Carlo means (standard deviations) are shown for TP, FP, FN, PPV and Se.

Graph	n	Algorithm	TP	FP	FN	PPV	Se
Scale-free	200	PC-LPGM	2.136 (1.617)	0.744 (0.927)	6.864 (1.617)	0.756 (0.267)	0.237 (0.180)
		LPGM	1.628 (1.249)	1.920 (1.885)	7.372 (1.249)	0.524 (0.336)	0.181 (0.139)
		PDN	3.824 (1.221)	4.200(1.655)	5.176 (1.221)	0.486 (0.164)	0.425 (0.136)
		VSL	1.934 (1.142)	0.658 (0.927)	7.066 (1.142)	0.797 (0.277)	0.215 (0.127)
		GLASSO	1.914 (1.119)	0.660 (0.937)	7.086 (1.119)	0.796 (0.278)	0.213 (0.124)
		NPN-Copula	2.012 (1.214)	0.550 (0.924)	6.988 (1.214)	0.840 (0.260)	0.224 (0.135)
		NPN-Skeptic	1.832 (1.302)	0.568 (0.927)	7.168 (1.302)	0.821 (0.237)	0.204 (0.145)
	1000	PC-LPGM	8.590 (0.764)	1.060 (0.926)	0.410 (0.764)	0.898 (0.084)	0.954 (0.085)
		LPGM	4.352 (1.818)	2.020 (1.699)	4.648 (1.818)	0.719 (0.198)	0.484 (0.202)
		PDN	6.148 (0.865)	0.366 (0.604)	2.852 (0.865)	0.948 (0.082)	0.683 (0.096)
		VSL	3.212 (1.742)	0.008 (0.089)	5.788 (1.742)	0.999 (0.015)	0.357 (0.194)
		GLASSO	3.194 (1.734)	0.008 (0.089)	5.806 (1.734)	0.997 (0.015)	0.355 (0.193)
		NPN-Copula	3.302 (1.722)	0.004 (0.063)	5.698 (1.722)	0.999 (0.017)	0.367 (0.191)
		NPN-Skeptic	3.058 (1.867)	0.004 (0.063)	5.942 (1.867)	0.999 (0.017)	0.340 (0.207)
	2000	PC-LPGM	8.996 (0.063)	1.118 (1.017)	0.004 (0.063)	0.898 (0.085)	1.000 (0.007)
		LPGM	4.828 (1.812)	2.320 (2.006)	4.172 (1.812)	0.720 (0.178)	0.536 (0.201)
		PDN	6.258 (0.803)	0.020 (0.140)	2.742 (0.803)	0.997 (0.020)	0.695 (0.089)
		VSL	4.238 (1.984)	0.000 (0.000)	4.762 (1.984)	1.000 (0.000)	0.471 (0.220)
		GLASSO	4.222 (1.975)	0.000 (0.000)	4.778 (1.975)	1.000 (0.000)	0.469 (0.219)
		NPN-Copula	4.408 (1.931)	0.000 (0.000)	4.592 (1.931)	1.000 (0.000)	0.490 (0.215)
		NPN-Skeptic	4.198 (2.102)	0.000 (0.000)	4.802 (2.102)	1.000 (0.000)	0.466 (0.234)
Hub	200	PC-LPGM	2.132 (1.535)	0.650 (0.830)	5.868 (1.535)	0.768 (0.278)	0.267 (0.192)
		LPGM	1.588 (1.363)	2.188 (2.212)	6.412 (1.363)	0.224 (0.334)	0.099 (0.170)
		PDN	3.366 (1.265)	4.876 (1.726)	4.634 (1.265)	0.416 (0.164)	0.421 (0.158)
		VSL	1.784 (1.002)	0.896 (1.236)	6.216 (1.002)	0.744 (0.300)	0.223 (0.125)
		GLASSO	1.766 (1.003)	0.890 (1.225)	6.234 (1.003)	0.744 (0.301)	0.221 (0.125)
		NPN-Copula	1.880 (1.073)	0.806 (1.109)	6.120 (1.073)	0.765 (0.297)	0.235 (0.134)
		NPN-Skeptic	1.694 (1.157)	0.842 (1.176)	6.306 (1.157)	0.738 (0.294)	0.212 (0.145)
	1000	PC-LPGM	7.608 (0.586)	1.150 (0.985)	0.392 (0.586)	0.879 (0.095)	0.951 (0.073)
		LPGM	4.268 (1.175)	2.636 (1.733)	3.732 (1.751)	0.636 (0.188)	0.534 (0.219)
		PDN	6.594 (0.864)	0.782 (0.914)	1.406 (0.864)	0.897 (0.116)	0.824 (0.108)
		VSL	3.152 (1.628)	0.012 (0.109)	4.848 (1.628)	1.000 (0.019)	0.394 (0.203)
		GLASSO	3.142 (1.620)	0.012 (0.109)	4.858 (1.620)	1.000 (0.019)	0.393 (0.202)
		NPN-Copula	3.168 (1.647)	0.006 (0.077)	4.832 (1.647)	1.000 (0.016)	0.396 (0.206)
		NPN-Skeptic	2.990 (1.737)	0.010 (0.100)	5.010 (1.737)	0.998 (0.021)	0.374 (0.217)
	2000	PC-LPGM	7.998 (0.045)	1.160 (0.998)	0.002 (0.045)	0.883 (0.092)	1.000 (0.006)
		LPGM	4.612 (2.231)	2.708 (1.901)	3.388 (2.231)	0.632 (0.234)	0.576 (0.279)
		PDN	7.158 (0.421)	0.046 (0.210)	0.842 (0.421)	0.994 (0.027)	0.895 (0.053)
		VSL	3.900 (1.823)	0.000 (0.000)	4.100 (1.823)	1.000 (0.000)	0.488 (0.228)
		GLASSO	3.874 (1.815)	0.000 (0.000)	4.126 (1.815)	1.000 (0.000)	0.484 (0.227)
		NPN-Copula	4.026 (1.881)	0.000 (0.000)	3.974 (1.881)	1.000 (0.000)	0.503 (0.235)
		NPN-Skeptic	3.730 (2.044)	0.000 (0.000)	4.270 (2.044)	1.000 (0.000)	0.466 (0.255)

Table B.2 – continued from previous page

Graph	n	Algorithm	TP	FP	FN	PPV	Se
Random	200	PC-LPGM	1.685 (1.437)	0.740 (0.973)	6.315 (1.437)	0.716 (0.305)	0.211 (0.180)
		LPGM	1.552 (1.189)	2.264 (2.553)	6.448 (1.189)	0.513 (0.349)	0.194 (0.149)
		PDN	3.204 (1.038)	4.904 (1.507)	4.796 (1.038)	0.402 (0.137)	0.400 (0.130)
		VSL	1.800 (1.103)	0.850 (1.295)	6.200 (1.103)	0.757 (0.310)	0.225 (0.138)
		GLASSO	1.805 (1.115)	0.845 (1.300)	6.195 (1.113)	0.758 (0.309)	0.226 (0.139)
		NPN-Copula	1.980 (1.194)	0.735 (1.184)	6.020 (1.194)	0.801 (0.281)	0.248 (0.149)
		NPN-Skeptic	1.795 (1.213)	0.830 (1.265)	6.205 (1.213)	0.752 (0.291)	0.224 (0.152)
	1000	LRTPC	7.470 (0.779)	0.980 (1.044)	0.530 (0.779)	0.895 (0.101)	0.934 (0.097)
		LPGM	3.724 (1.660)	1.872 (1.850)	4.276 (1.660)	0.704 (0.250)	0.466 (0.207)
		PDN	4.816 (0.709)	2.600 (0.823)	3.184 (0.709)	0.653 (0.081)	0.602 (0.089)
		VSL	3.042 (1.588)	0.016 (0.126)	4.958 (1.588)	0.997 (0.027)	0.380 (0.198)
		GLASSO	3.018 (1.563)	0.016 (0.126)	4.982 (1.563)	0.997 (0.027)	0.377 (0.195)
		NPN-Copula	3.164 (1.588)	0.008 (0.089)	4.836 (1.588)	0.998 (0.017)	0.396 (0.199)
		NPN-Skeptic	2.972 (1.699)	0.010 (0.100)	5.028 (1.699)	0.998 (0.022)	0.372 (0.212)
	2000	LRTPC	8.000 (0.000)	0.848 (0.944)	0.000 (0.000)	0.914 (0.089)	1.000 (0.000)
		LPGM	3.572 (1.533)	1.424 (1.304)	4.428 (1.533)	0.758 (0.191)	0.446 (0.192)
		PDN	5.044 (0.732)	2.348 (0.645)	2.956 (0.732)	0.685 (0.065)	0.630 (0.091)
		VSL	3.665 (1.803)	0.000 (0.000)	4.335 (1.803)	1.000 (0.000)	0.458 (0.225)
		GLASSO	3.640 (1.791)	0.000 (0.000)	4.360 (1.791)	1.000 (0.000)	0.455 (0.224)
		NPN-Copula	3.785 (1.823)	0.000 (0.000)	4.215 (1.823)	1.000 (0.000)	0.473 (0.228)
		NPN-Skeptic	3.610 (2.044)	0.000 (0.000)	4.390 (2.044)	1.000 (0.000)	0.451 (0.256)

Table B.3: Simulation results from 500 replicates of the undirected graphs shown in Figure 3.2 for $p = 100$ variables with Poisson node conditional distribution and level of noise $\lambda_{noise} = 0.5$. Monte Carlo means (standard deviations) are shown for TP, FP, FN, PPV and Se.

Graph	n	Algorithm	TP	FP	FN	PPV	Se
Scale-free	200	PC-LPGM	61.585 (4.316)	8.490 (2.887)	37.415 (4.216)	0.880 (0.038)	0.622 (0.044)
		LPGM	5.564 (8.084)	0.824 (5.594)	93.436 (8.084)	0.985 (0.067)	0.056 (0.082)
		PDN	53.080 (3.283)	26.007 (4.942)	45.920 (3.283)	0.673 (0.052)	0.536 (0.033)
		VSL	63.915 (6.489)	22.308 (13.433)	35.085 (6.489)	0.760 (0.095)	0.646 (0.066)
		GLASSO	62.755 (6.306)	22.642 (13.114)	36.245 (6.306)	0.754 (0.097)	0.634 (0.064)
		NPN-Copula	65.647 (5.734)	18.345 (11.701)	33.352 (5.734)	0.797 (0.088)	0.663 (0.058)
		NPN-Skeptic	64.343 (6.316)	22.918 (15.323)	34.657 (6.316)	0.759 (0.102)	0.650 (0.064)
	1000	PC-LPGM	98.580 (0.610)	9.589 (2.982)	0.420 (0.610)	0.912 (0.025)	0.996 (0.006)
		LPGM	51.520 (11.263)	0.012 (0.109)	47.480 (11.263)	1.000 (0.002)	0.520 (0.114)
		PDN	65.357 (1.871)	0.050 (0.218)	33.643 (1.871)	0.999 (0.003)	0.660 (0.019)
		VSL	94.438 (2.316)	0.089 (0.286)	4.562 (2.316)	0.999 (0.003)	0.954 (0.023)
		GLASSO	93.830 (2.507)	0.161 (0.393)	5.170 (2.507)	0.998 (0.004)	0.948 (0.025)
		NPN-Copula	94.571 (2.159)	0.054 (0.226)	4.429 (2.159)	1.000 (0.002)	0.955 (0.022)
		NPN-Skeptic	94.277 (2.089)	0.134 (0.342)	4.723 (2.089)	0.999 (0.004)	0.952 (0.021)

Table B.3 – continued from previous page

Graph	n	Algorithm	TP	FP	FN	PPV	Se
	2000	PC-LPGMC	99.000 (0.000)	9.759 (3.134)	0.000 (0.000)	0.911 (0.026)	1.000 (0.000)
		LPGM	54.185 (2.379)	0.010 (0.100)	44.815 (2.379)	1.000 (0.002)	0.547 (0.024)
		PDN	64.370 (1.560)	0.000 (0.000)	34.630 (1.560)	1.000 (0.000)	0.650 (0.016)
		VSL	96.821 (1.422)	0.000 (0.000)	2.179 (1.422)	1.000 (0.000)	0.978 (0.014)
		GLASSO	96.518 (1.577)	0.000 (0.000)	2.482 (1.577)	1.000 (0.000)	0.975 (0.016)
		NPN-Copula	97.375 (1.402)	0.000 (0.000)	1.625 (1.402)	1.000 (0.000)	0.984 (0.014)
		NPN-Skeptic	97.214 (1.423)	0.009 (0.000)	1.786 (1.423)	1.000 (0.000)	0.982 (0.014)
Hub	200	PC-LPGM	13.393 (2.484)	14.518 (4.082)	81.607 (2.484)	0.486 (0.084)	0.141 (0.026)
		LPGM	4.344 (4.368)	5.840 (9.239)	90.656 (4.368)	0.426 (0.330)	0.046 (0.046)
		PDN	19.340 (3.834)	84.747 (5.935)	75.660 (3.834)	0.186 (0.038)	0.204 (0.040)
		VSL	16.643 (6.546)	26.982 (17.330)	78.357 (6.546)	0.427 (0.128)	0.175 (0.069)
		GLASSO	15.991 (6.361)	25.518 (16.665)	79.009 (6.361)	0.434 (0.135)	0.168 (0.067)
		NPN-Copula	18.491 (6.864)	26.625 (16.889)	76.509 (6.864)	0.451 (0.121)	0.195 (0.072)
		NPN-Skeptic	17.473 (7.408)	31.348 (22.170)	77.527 (7.408)	0.406 (0.123)	0.184 (0.078)
	1000	PC-LPGM	84.794 (3.416)	25.238 (5.079)	10.206 (3.416)	0.772 (0.036)	0.893 (0.036)
		LPGM	4.555 (6.512)	0.910 (1.349)	90.445 (6.512)	0.792 (0.324)	0.048 (0.069)
		PDN	78.487 (3.585)	19.650 (4.209)	16.513 (3.585)	0.800 (0.041)	0.826 (0.038)
		VSL	29.651 (12.504)	0.063 (0.303)	65.349 (12.504)	0.998 (0.010)	0.312 (0.132)
		GLASSO	29.341 (12.233)	0.056 (0.262)	65.659 (12.233)	0.998 (0.009)	0.309 (0.129)
		NPN-Copula	37.746 (15.112)	0.048 (0.248)	57.254 (15.112)	0.999 (0.004)	0.397 (0.159)
		NPN-Skeptic	35.476 (16.277)	0.119 (0.412)	59.524 (16.277)	0.998 (0.007)	0.373 (0.171)
	2000	PC-LPGM	94.949 (0.221)	26.942 (5.566)	0.051 (0.221)	0.781 (0.036)	0.999 (0.002)
		LPGM	7.145 (9.369)	0.625 (0.805)	87.855 (9.369)	0.620 (0.478)	0.075 (0.099)
		PDN	93.073 (1.205)	1.113 (1.094)	1.927 (1.205)	0.988 (0.012)	0.980 (0.013)
		VSL	69.263 (15.639)	0.013 (0.113)	25.737 (15.639)	1.000 (0.001)	0.729 (0.165)
		GLASSO	68.647 (14.931)	0.013 (0.113)	26.353 (14.931)	1.000 (0.001)	0.723 (0.157)
		NPN-Copula	77.833 (8.985)	0.000 (0.000)	17.167 (8.895)	1.000 (0.000)	0.819 (0.095)
		NPN-Skeptic	74.987 (9.809)	0.013 (0.000)	20.013 (8.985)	1.000 (0.001)	0.789 (0.103)
Random	200	PC-LPGM	62.432 (5.030)	8.656 (2.998)	46.568 (5.030)	0.879 (0.039)	0.573 (0.046)
		LPGM	8.190 (2.370)	0.120 (0.326)	100.810 (2.370)	0.987 (0.036)	0.075 (0.025)
		PDN	52.007 (3.302)	32.167 (5.283)	56.993 (3.302)	0.619 (0.049)	0.477 (0.030)
		VSL	67.032 (8.241)	26.932 (15.060)	41.968 (8.241)	0.735 (0.100)	0.615 (0.076)
		GLASSO	64.736 (8.543)	25.440 (15.001)	44.264 (8.543)	0.742 (0.106)	0.594 (0.078)
		NPN-Copula	70.520 (7.514)	23.344 (13.387)	38.480 (7.514)	0.769 (0.091)	0.647 (0.069)
		NPN-Skeptic	68.956 (8.123)	29.956 (18.522)	40.044 (8.123)	0.722 (0.105)	0.633 (0.075)
	1000	PC-LPGM	105.748 (1.504)	8.752 (2.939)	3.252 (1.504)	0.924 (0.024)	0.970 (0.014)
		LPGM	43.800 (31.795)	0.300 (0.593)	65.200 (31.795)	0.996 (0.009)	0.402 (0.292)
		PDN	63.020 (2.491)	9.470 (1.332)	45.980 (2.491)	0.870 (0.016)	0.578 (0.023)
		VSL	102.676 (3.506)	0.136 (4.123)	6.324 (3.506)	0.999 (0.003)	0.942 (0.032)
		GLASSO	101.904 (4.123)	0.152 (0.142)	7.096 (4.123)	0.999 (0.004)	0.935 (0.038)
		NPN-Copula	104.820 (2.159)	0.104 (0.319)	4.180 (2.159)	0.999 (0.003)	0.962 (0.020)
		NPN-Skeptic	104.392 (2.237)	0.192 (0.424)	4.608 (2.237)	0.998 (0.004)	0.958 (0.021)
	2000	PC-LPGM	106.724 (1.212)	8.664 (2.855)	2.276 (1.212)	0.925 (0.023)	0.979 (0.011)
		LPGM	69.900 (7.493)	0.280 (0.577)	39.100 (7.493)	0.996 (0.008)	0.641 (0.069)

Table B.3 – continued from previous page

Graph	n	Algorithm	TP	FP	FN	PPV	Se
		PDN	62.850 (2.243)	9.230 (1.439)	46.150 (2.243)	0.872 (0.018)	0.577 (0.021)
		VSL	106.836 (1.365)	0.000 (0.000)	2.164 (1.365)	1.000 (0.000)	0.980 (0.013)
		GLASSO	106.884 (1.350)	0.000 (0.000)	2.116 (1.350)	1.000 (0.000)	0.981 (0.012)
		NPN-Copula	107.376 (1.253)	0.000 (0.000)	1.624 (1.253)	1.000 (0.000)	0.985 (0.011)
		NPN-Skeptic	107.124 (1.322)	0.000 (0.000)	1.876 (1.322)	1.000 (0.000)	0.983 (0.012)

Table B.4: Simulation results from 500 replicates of the undirected graphs shown in Figure 3.2 for $p = 100$ variables with Poisson node conditional distribution and level of noise $\lambda_{noise} = 5$. Monte Carlo means (standard deviations) are shown for TP, FP, FN, PPV and Se.

Graph	n	Algorithm	TP	FP	FN	PPV	Se
Scale-free	200	PC-LPGM	7.780 (2.843)	14.470 (3.705)	91.220 (2.843)	0.348 (0.100)	0.079 (0.029)
		LPGM	10.188 (4.126)	65.352 (20.496)	88.812 (4.126)	0.152 (0.127)	0.103 (0.042)
		PDN	13.457 (3.164)	94.817 (6.073)	85.543 (3.164)	0.125 (0.030)	0.136 (0.032)
		VSL	9.316 (4.895)	22.496 (16.852)	89.684 (4.895)	0.332 (0.119)	0.094 (0.049)
		GLASSO	9.052 (4.775)	21.372 (16.016)	89.948 (4.775)	0.336 (0.120)	0.091 (0.048)
		NPN-Copula	10.012 (5.255)	21.924 (16.439)	88.988 (5.255)	0.359 (0.135)	0.101 (0.053)
		NPN-Skeptic	9.868 (5.979)	27.424 (24.698)	89.132 (5.979)	0.320 (0.132)	0.100 (0.060)
	1000	PC-LPGM	75.130 (4.420)	24.805 (4.647)	23.870 (4.420)	0.753 (0.038)	0.759 (0.045)
		LPGM	1.480 (1.696)	1.892 (3.146)	97.520 (1.696)	0.574 (0.412)	0.015 (0.017)
		PDN	52.827 (3.386)	31.153 (5.108)	46.173 (3.386)	0.630 (0.049)	0.534 (0.034)
		VSL	14.844 (6.389)	0.044 (0.224)	84.156 (6.389)	0.998 (0.013)	0.150 (0.065)
		GLASSO	14.936 (6.455)	0.044 (0.224)	84.064 (6.455)	0.998 (0.013)	0.151 (0.065)
		NPN-Copula	17.124 (7.494)	0.040 (0.196)	81.876 (7.494)	0.998 (0.009)	0.173 (0.076)
		NPN-Skeptic	16.708 (8.088)	0.116 (0.419)	82.292 (8.088)	0.996 (0.014)	0.169 (0.082)
	2000	PC-LPGMC	96.400 (1.515)	26.500 (5.147)	2.600 (1.514)	0.786 (0.033)	0.974 (0.015)
		LPGM	2.800 (2.138)	1.004 (1.455)	96.200 (2.138)	0.785 (0.266)	0.028 (0.022)
		PDN	67.917 (2.591)	4.413 (2.140)	31.083 (2.591)	0.939 (0.029)	0.686 (0.026)
		VSL	24.579 (11.580)	0.000 (0.000)	74.421 (11.580)	1.000 (0.000)	0.255 (0.117)
		GLASSO	25.733 (12.171)	0.000 (0.000)	73.267 (12.171)	1.000 (0.000)	0.264 (0.123)
		NPN-Copula	33.672 (14.879)	0.000 (0.000)	65.328 (14.879)	1.000 (0.000)	0.335 (0.150)
		NPN-Skeptic	32.267 (15.750)	0.000 (0.000)	66.733 (15.750)	1.000 (0.000)	0.321 (0.159)
	200	PC-LPGM	2.690 (1.705)	13.600 (4.476)	92.310 (1.705)	0.166 (0.101)	0.028 (0.018)
		LPGM	0.444 (1.175)	34.632 (33.612)	94.556 (1.175)	0.046 (0.152)	0.005 (0.012)
		PDN	6.630 (2.373)	103.063 (4.902)	88.370 (2.373)	0.060 (0.021)	0.070 (0.025)
		VSL	3.392 (2.233)	23.688 (15.017)	91.608 (2.233)	0.143 (0.097)	0.036 (0.024)
		GLASSO	3.304 (2.139)	22.964 (14.511)	91.696 (2.139)	0.145 (0.099)	0.035 (0.023)
		NPN-Copula	3.392 (2.189)	21.852 (13.797)	91.608 (2.189)	0.150 (0.097)	0.036 (0.023)
		NPN-Skeptic	3.108 (2.297)	23.476 (19.474)	91.892 (2.297)	0.134 (0.091)	0.033 (0.024)
	1000	PC-LPGM	29.525 (3.837)	24.635 (5.206)	65.475 (3.837)	0.548 (0.029)	0.311 (0.020)
		LPGM	0.892 (2.246)	1.076 (2.639)	94.108 (2.246)	0.439 (0.389)	0.009 (0.012)

Table B.4 – continued from previous page

Graph	n	Algorithm	TP	FP	FN	PPV	Se
Hub		PDN	23.427 (3.516)	84.433 (5.305)	71.573 (3.316)	0.217 (0.033)	0.247 (0.037)
		VSL	7.424 (4.075)	1.428 (2.091)	87.576 (4.075)	0.884 (0.137)	0.078 (0.043)
		GLASSO	7.364 (4.053)	1.424 (2.095)	87.636 (4.053)	0.883 (0.138)	0.078 (0.043)
		NPN-Copula	8.440 (4.399)	1.392 (2.018)	86.560 (4.399)	0.895 (0.126)	0.089 (0.046)
		NPN-Skeptic	8.208 (4.629)	1.804 (2.291)	86.792 (4.629)	0.860 (0.134)	0.086 (0.049)
	2000	PC-LPGM	65.025 (4.253)	29.855 (5.473)	29.975 (4.253)	0.687 (0.041)	0.684 (0.045)
		LPGM	0.392 (0.796)	1.712 (1.971)	94.608 (0.796)	0.187 (0.339)	0.004 (0.008)
		PDN	49.100 (4.566)	54.997 (5.883)	45.900 (4.566)	0.472 (0.047)	0.517 (0.048)
		VSL	8.983 (6.782)	0.068 (0.284)	86.017 (6.782)	0.996 (0.018)	0.095 (0.071)
		GLASSO	8.924 (6.748)	0.068 (0.284)	86.076 (6.748)	0.996 (0.018)	0.094 (0.071)
		NPN-Copula	9.797 (7.547)	0.042 (0.241)	85.203 (7.547)	0.998 (0.012)	0.103 (0.079)
		NPN-Skeptic	9.305 (7.052)	0.068 (0.284)	85.695 (7.052)	0.995 (0.020)	0.098 (0.074)
Random	200	PC-LPGM	8.040 (2.884)	14.805 (3.878)	100.960 (2.884)	0.350 (0.093)	0.074 (0.026)
		LPGM	10.592 (4.318)	69.316 (25.767)	98.408 (4.318)	0.175 (0.189)	0.097 (0.040)
		PDN	13.573 (2.989)	94.750 (5.987)	95.427 (2.989)	0.126 (0.029)	0.125 (0.027)
		VSL	10.548 (5.213)	22.848 (15.701)	98.452 (5.213)	0.353 (0.119)	0.097 (0.048)
		GLASSO	10.160 (5.075)	21.460 (14.871)	98.840 (5.075)	0.358 (0.119)	0.093 (0.047)
		NPN-Copula	11.064 (5.327)	22.136 (16.599)	97.936 (5.327)	0.382 (0.131)	0.102 (0.049)
		NPN-Skeptic	10.648 (6.242)	26.632 (23.376)	98.352 (6.642)	0.341 (0.134)	0.098 (0.057)
	1000	PC-LPGM	81.055 (4.632)	23.665 (4.941)	27.945 (4.632)	0.775 (0.038)	0.744 (0.042)
		LPGM	1.776 (2.675)	3.196 (5.107)	107.224 (2.675)	0.397 (0.401)	0.016 (0.025)
		PDN	53.207 (3.471)	33.383 (10.084)	55.793 (3.471)	0.616 (0.046)	0.488 (0.032)
		VSL	14.741 (6.294)	0.022 (0.148)	94.259 (6.294)	0.999 (0.006)	0.135 (0.058)
		GLASSO	14.741 (6.291)	0.022 (0.148)	94.259 (6.291)	0.999 (0.006)	0.135 (0.058)
		NPN-Copula	16.333 (7.249)	0.022 (0.148)	92.667 (7.249)	0.999 (0.005)	0.150 (0.067)
		NPN-Skeptic	15.178 (7.307)	0.044 (0.296)	93.822 (7.307)	0.998 (0.011)	0.139 (0.067)
	2000	PC-LPGM	104.010 (1.992)	24.370 (4.706)	4.990 (1.992)	0.811 (0.029)	0.954 (0.018)
		LPGM	1.995 (1.800)	1.260 (1.825)	107.005 (1.880)	0.671 (0.360)	0.018 (0.017)
		PDN	65.093 (2.892)	12.297 (1.837)	43.907 (2.892)	0.841 (0.021)	0.597 (0.027)
		VSL	26.038 (12.457)	0.000 (0.000)	82.962 (12.457)	1.000 (0.000)	0.239 (0.114)
		GLASSO	26.327 (12.487)	0.000 (0.000)	82.673 (12.487)	1.000 (0.000)	0.242 (0.115)
		NPN-Copula	30.340 (14.496)	0.000 (0.000)	78.660 (14.496)	1.000 (0.000)	0.278 (0.133)
		NPN-Skeptic	28.474 (14.777)	0.000 (0.000)	80.526 (14.777)	1.000 (0.000)	0.261 (0.136)

B.2 Appendix B.2

Table B.5, and Table B.6 report TP, FP, FN, PPV and Se for each of methods considered in Section 4.4. Two different graph dimensions, i.e., $p = 10, 100$, and three graph structures (see Figure 4.1 and Figure 4.2) are considered.

Table B.5: Simulation results from 500 replicates of the DAGs shown in Figure 4.1 for $p = 10$ variables with Poisson node conditional distribution. Monte Carlo means (standard deviations) are shown for TP, FP, FN, PPV and Se.

Graph	n	Algorithm	TP	FP	FN	PPV	Se
Scale-free	200	PKBIC	7.660 (0.913)	0.776 (0.869)	1.340 (0.913)	0.916 (0.090)	0.851 (0.101)
		PKAIC	8.364 (0.648)	4.632 (1.757)	0.636 (0.648)	0.654 (0.095)	0.929 (0.072)
		Or-PPGM	6.898 (0.939)	0.270 (0.519)	2.102 (0.939)	0.966 (0.064)	0.766 (0.104)
		Or-LPGM	8.550 (0.679)	6.872 (3.155)	0.450 (0.679)	0.577 (0.119)	0.950 (0.075)
		PDN	3.600 (0.688)	5.628 (0.680)	5.400 (0.688)	0.390 (0.068)	0.400 (0.076)
		ODS	4.724 (1.100)	10.722 (3.344)	4.276 (1.100)	0.318 (0.096)	0.525 (0.122)
		MMHC	2.988 (1.120)	4.374 (1.309)	6.012 (1.120)	0.407 (0.143)	0.332 (0.124)
		K2mix	5.138 (1.125)	0.732 (0.891)	3.862 (1.125)	0.890 (0.127)	0.571 (0.125)
		K2cut	3.094 (0.819)	0.348 (0.619)	5.906 (0.819)	0.917 (0.139)	0.344 (0.091)
		PCmix	2.766 (0.968)	0.872 (0.798)	6.234 (0.968)	0.772 (0.207)	0.307 (0.108)
		PClog	4.314 (0.868)	2.094 (0.918)	4.686 (0.868)	0.678 (0.114)	0.479 (0.096)
	1000	PKBIC	8.970 (0.171)	0.276 (0.518)	0.030 (0.171)	0.973 (0.050)	0.997 (0.019)
		PKAIC	8.992 (0.089)	4.134 (1.688)	0.008 (0.089)	0.697 (0.092)	0.999 (0.010)
		Or-PPGM	8.930 (0.255)	0.238 (0.500)	0.070 (0.255)	0.977 (0.048)	0.992 (0.028)
		Or-LPGM	8.998 (0.045)	2.098 (1.559)	0.002 (0.045)	0.826 (0.108)	1.000 (0.005)
		PDN	3.242 (0.477)	5.904 (0.308)	5.758 (0.477)	0.353 (0.040)	0.360 (0.053)
		ODS	6.274 (1.081)	5.704 (2.017)	2.726 (1.081)	0.534 (0.121)	0.697 (0.120)
		MMHC	4.898 (1.242)	4.294 (1.221)	4.102 (1.242)	0.532 (0.130)	0.544 (0.138)
		K2mix	7.716 (0.465)	0.012 (0.109)	1.284 (0.465)	0.999 (0.013)	0.857 (0.052)
		K2cut	4.444 (0.616)	0.062 (0.273)	4.556 (0.616)	0.989 (0.047)	0.494 (0.068)
		PCmix	5.458 (0.749)	2.244 (0.719)	3.542 (0.749)	0.710 (0.085)	0.606 (0.083)
		PClog	6.544 (0.708)	2.434 (0.836)	2.456 (0.708)	0.731 (0.082)	0.727 (0.079)
	2000	PKBIC	9.000 (0.000)	0.162 (0.405)	0.000 (0.000)	0.984 (0.039)	1.000 (0.000)
		PKAIC	9.000 (0.000)	4.310 (1.699)	0.000 (0.000)	0.687 (0.089)	1.000 (0.000)
		Or-PPGM	8.998 (0.045)	0.230 (0.492)	0.002 (0.045)	0.978 (0.047)	1.000 (0.005)
		Or-LPGM	9.000 (0.000)	0.702 (0.905)	0.000 (0.000)	0.935 (0.078)	1.000 (0.000)
		PDN	3.110 (0.320)	5.986 (0.118)	5.890 (0.320)	0.341 (0.023)	0.346 (0.036)
		ODS	6.856 (0.989)	3.922 (1.569)	2.144 (0.989)	0.644 (0.120)	0.762 (0.110)
		MMHC	5.534 (0.855)	3.758 (1.038)	3.466 (0.855)	0.598 (0.098)	0.615 (0.095)
		K2mix	7.776 (0.417)	0.004 (0.063)	1.224 (0.417)	1.000 (0.007)	0.864 (0.046)
		K2cut	4.860 (0.679)	0.014 (0.118)	4.140 (0.679)	0.997 (0.023)	0.540 (0.075)
		PCmix	6.378 (0.651)	2.212 (0.551)	2.622 (0.651)	0.743 (0.060)	0.709 (0.072)
		PClog	6.770 (0.618)	2.356 (0.814)	2.230 (0.618)	0.744 (0.077)	0.752 (0.069)
	200	PKBIC	3.366 (0.722)	0.782 (0.885)	4.634 (0.722)	0.837 (0.167)	0.421 (0.090)
		PKAIC	4.132 (0.761)	5.098 (2.084)	3.868 (0.761)	0.469 (0.129)	0.516 (0.095)
		Or-PPGM	3.814 (0.785)	0.378 (0.678)	4.186 (0.785)	0.924 (0.128)	0.477 (0.098)
		Or-LPGM	4.614 (0.900)	5.150 (2.873)	3.386 (0.900)	0.513 (0.156)	0.577 (0.113)
		PDN	1.928 (0.315)	7.548 (0.904)	6.072 (0.315)	0.205 (0.035)	0.241 (0.039)
		ODS	2.560 (1.235)	6.882 (3.167)	5.440 (1.235)	0.291 (0.153)	0.320 (0.154)
		MMHC	1.116 (0.464)	2.780 (1.194)	6.884 (0.464)	0.309 (0.125)	0.140 (0.058)
		K2mix	1.960 (0.554)	0.736 (0.934)	6.040 (0.554)	0.793 (0.233)	0.245 (0.069)
		K2cut	1.074 (0.348)	0.448 (0.651)	6.926 (0.348)	0.800 (0.268)	0.134 (0.043)
		PCmix	0.802 (0.596)	1.192 (0.490)	7.198 (0.596)	0.356 (0.241)	0.100 (0.075)
		PClog	1.644 (0.786)	1.346 (0.734)	6.356 (0.786)	0.541 (0.190)	0.206 (0.098)

Table B.5 – continued from previous page

Graph	n	Algorithm	TP	FP	FN	PPV	Se
Hub	1000	PKBIC	4.020 (0.140)	0.260 (0.507)	3.980 (0.140)	0.950 (0.094)	0.502 (0.018)
		PKAIC	4.446 (0.610)	4.978 (2.008)	3.554 (0.610)	0.494 (0.125)	0.556 (0.076)
		Or-PPGM	5.794 (0.443)	0.342 (0.631)	2.206 (0.443)	0.952 (0.084)	0.724 (0.055)
		Or-LPGM	5.038 (0.317)	1.432 (1.278)	2.962 (0.317)	0.807 (0.146)	0.630 (0.040)
		PDN	2.018 (0.133)	7.152 (0.744)	5.982 (0.133)	0.221 (0.020)	0.252 (0.017)
		ODS	3.406 (0.916)	3.354 (1.824)	4.594 (0.916)	0.530 (0.188)	0.426 (0.114)
		MMHC	1.146 (0.661)	3.878 (0.932)	6.854 (0.661)	0.230 (0.113)	0.143 (0.083)
		K2mix	2.406 (0.500)	0.028 (0.165)	5.594 (0.500)	0.992 (0.048)	0.301 (0.062)
		K2cut	1.416 (0.509)	0.080 (0.279)	6.584 (0.509)	0.966 (0.120)	0.177 (0.064)
		PCmix	2.320 (0.698)	1.114 (0.407)	5.680 (0.698)	0.664 (0.120)	0.290 (0.087)
		PClog	3.560 (0.629)	1.302 (0.687)	4.440 (0.629)	0.737 (0.111)	0.445 (0.079)
	2000	PKBIC	4.016 (0.126)	0.172 (0.408)	3.984 (0.126)	0.967 (0.078)	0.502 (0.016)
		PKAIC	4.498 (0.612)	4.972 (1.981)	3.502 (0.612)	0.495 (0.118)	0.562 (0.077)
		Or-PPGM	5.992 (0.167)	0.380 (0.693)	2.008 (0.167)	0.950 (0.087)	0.749 (0.021)
		Or-LPGM	5.010 (0.118)	0.454 (0.649)	2.990 (0.118)	0.928 (0.098)	0.626 (0.015)
		PDN	2.004 (0.063)	7.050 (0.823)	5.996 (0.063)	0.223 (0.021)	0.250 (0.008)
		ODS	3.824 (0.794)	1.884 (1.234)	4.176 (0.794)	0.686 (0.181)	0.478 (0.099)
		MMHC	1.074 (0.538)	4.130 (0.752)	6.926 (0.538)	0.207 (0.084)	0.134 (0.067)
		K2mix	2.954 (0.237)	0.018 (0.133)	5.046 (0.237)	0.995 (0.035)	0.369 (0.030)
		K2cut	1.626 (0.509)	0.016 (0.126)	6.374 (0.509)	0.994 (0.048)	0.203 (0.064)
		PCmix	2.988 (0.465)	1.116 (0.432)	5.012 (0.465)	0.729 (0.081)	0.374 (0.058)
		PClog	3.908 (0.390)	1.274 (0.663)	4.092 (0.390)	0.761 (0.093)	0.488 (0.049)
Random	200	PKBIC	3.338 (0.999)	0.780 (0.866)	4.662 (0.999)	0.834 (0.172)	0.417 (0.125)
		PKAIC	4.872 (0.979)	5.198 (1.974)	3.128 (0.979)	0.498 (0.122)	0.609 (0.122)
		Or-PPGM	2.680 (0.907)	0.508 (0.689)	5.320 (0.907)	0.868 (0.173)	0.335 (0.113)
		Or-LPGM	4.778 (1.213)	4.934 (2.739)	3.222 (1.213)	0.523 (0.145)	0.597 (0.152)
		PDN	2.406 (0.728)	7.238 (1.023)	5.594 (0.728)	0.250 (0.075)	0.301 (0.091)
		ODS	2.390 (1.249)	7.250 (3.219)	5.610 (1.249)	0.268 (0.156)	0.299 (0.156)
		MMHC	2.454 (0.987)	1.312 (1.135)	5.546 (0.987)	0.689 (0.244)	0.307 (0.123)
		CK2mix	1.854 (0.826)	0.936 (1.034)	6.146 (0.826)	0.721 (0.261)	0.232 (0.103)
		CK2cut	1.034 (0.601)	0.504 (0.704)	6.966 (0.601)	0.743 (0.327)	0.129 (0.075)
		PCmix	0.144 (0.490)	1.418 (0.800)	7.856 (0.490)	0.066 (0.219)	0.018 (0.061)
		PClog	1.076 (1.195)	1.668 (0.918)	6.924 (1.195)	0.331 (0.361)	0.134 (0.149)
	1000	PKBIC	5.326 (0.661)	0.290 (0.524)	2.674 (0.661)	0.955 (0.080)	0.666 (0.083)
		PKAIC	6.130 (0.628)	4.814 (1.998)	1.870 (0.628)	0.578 (0.117)	0.766 (0.078)
		Or-PPGM	5.256 (0.684)	0.362 (0.645)	2.744 (0.684)	0.946 (0.091)	0.657 (0.085)
		Or-LPGM	5.874 (0.554)	1.308 (1.271)	2.126 (0.554)	0.841 (0.133)	0.734 (0.069)
		PDN	2.974 (0.171)	6.292 (0.729)	5.026 (0.171)	0.323 (0.031)	0.372 (0.021)
		ODS	3.402 (1.099)	4.100 (1.827)	4.598 (1.099)	0.469 (0.171)	0.425 (0.137)
		MMHC	4.352 (0.889)	1.096 (1.038)	3.648 (0.889)	0.811 (0.169)	0.544 (0.111)
		K2mix	3.102 (0.687)	0.052 (0.222)	4.898 (0.687)	0.987 (0.055)	0.388 (0.086)
		K2cut	1.690 (0.683)	0.072 (0.274)	6.310 (0.683)	0.972 (0.107)	0.211 (0.085)
		PCmix	2.034 (1.003)	1.618 (0.760)	5.966 (1.003)	0.541 (0.239)	0.254 (0.125)
		PClog	3.860 (1.088)	1.398 (0.597)	4.140 (1.088)	0.723 (0.137)	0.482(0.136)
	2000	PKBIC	5.882 (0.369)	0.214 (0.482)	2.118 (0.369)	0.970 (0.067)	0.735 (0.046)
		PKAIC	6.308 (0.523)	4.990 (1.990)	1.692 (0.523)	0.575 (0.107)	0.788 (0.065)

Table B.5 – continued from previous page

Graph	n	Algorithm	TP	FP	FN	PPV	Se
		Or-PPGM	5.894 (0.378)	0.544 (0.694)	2.106 (0.378)	0.925 (0.091)	0.737 (0.047)
		Or-LPGM	5.960 (0.266)	0.354 (0.581)	2.040 (0.266)	0.951 (0.078)	0.745 (0.033)
		PDN	2.994 (0.077)	6.156 (0.664)	5.006 (0.077)	0.329 (0.025)	0.374 (0.010)
		ODS	3.884 (1.132)	2.684 (1.517)	4.116 (1.132)	0.604 (0.196)	0.486 (0.141)
		MMHC	5.012 (0.840)	0.998 (0.932)	2.988 (0.840)	0.840 (0.140)	0.626 (0.105)
		K2mix	3.880 (0.407)	0.022 (0.147)	4.120 (0.407)	0.996 (0.030)	0.485 (0.051)
		K2cut	2.274 (0.696)	0.026 (0.159)	5.726 (0.696)	0.992 (0.052)	0.284 (0.087)
		PCmix	2.872 (1.072)	1.638 (0.639)	5.128 (1.072)	0.622 (0.165)	0.359 (0.134)
		PClog	4.808 (0.590)	1.228 (0.465)	3.192 (0.590)	0.798 (0.066)	0.601 (0.074)

Table B.6: Simulation results from 500 replicates of the DAGs shown in Figure 4.2 for $p = 100$ variables with Poisson node conditional distribution. Monte Carlo means (standard deviations) are shown for TP, FP, FN, PPV and Se.

Graph	n	Algorithm	TP	FP	FN	PPV	Se
Scale-free	200	PKBIC	66.870 (2.914)	74.880 (7.005)	32.130 (2.914)	0.473 (0.027)	0.675 (0.029)
		Or-PPGM	61.836 (2.944)	87.492 (4.288)	37.164 (2.944)	0.414 (0.019)	0.625 (0.030)
		Or-LPGM	68.206 (3.438)	118.400 (21.501)	30.794 (3.438)	0.370 (0.040)	0.689 (0.035)
		PDN	26.572 (5.971)	72.228 (21.678)	72.428 (5.971)	0.299 (0.123)	0.268 (0.060)
		ODS	33.782 (4.472)	152.354 (22.276)	65.218 (4.472)	0.184 (0.032)	0.341 (0.045)
		MMHC	32.828 (4.439)	89.412 (7.336)	66.172 (4.439)	0.269 (0.037)	0.332 (0.045)
		K2mix	39.004 (3.204)	95.964 (9.510)	59.996 (3.204)	0.290 (0.029)	0.394 (0.032)
		PClog	29.180 (3.413)	39.388 (4.964)	69.820 (3.413)	0.427 (0.051)	0.295 (0.034)
	1000	PKBIC	83.554 (1.127)	33.038 (5.158)	15.446 (1.127)	0.718 (0.032)	0.844 (0.011)
		Or-PPGM	87.610 (1.376)	81.994 (4.577)	11.390 (1.376)	0.517 (0.015)	0.885 (0.014)
		Or-LPGM	85.764 (1.689)	17.778 (5.895)	13.236 (1.689)	0.831 (0.046)	0.866 (0.017)
		PDN	29.558 (14.034)	39.488 (22.757)	69.442 (14.034)	0.496 (0.160)	0.299 (0.142)
		ODS	53.910 (3.960)	64.874 (9.449)	45.090 (3.960)	0.456 (0.048)	0.545 (0.040)
		MMHC	61.954 (4.035)	69.644 (7.537)	37.046 (4.035)	0.472 (0.037)	0.626 (0.041)
		K2mix	65.576 (2.246)	6.522 (3.992)	33.424 (2.246)	0.912 (0.050)	0.662 (0.023)
		PClog	48.790 (4.003)	53.670 (6.318)	50.210 (4.003)	0.477 (0.047)	0.493 (0.040)
	2000	PKBIC	84.870 (0.462)	22.912 (4.482)	14.130 (0.462)	0.789 (0.033)	0.857 (0.005)
		Or-PPGM	89.400 (5.702)	80.352 (6.891)	9.204 (0.834)	0.525 (0.036)	0.903 (0.058)
		Or-LPGM	88.454 (1.105)	3.647 (2.521)	10.546 (1.105)	0.961 (0.025)	0.893 (0.011)
		PDN	25.432 (15.366)	28.634 (20.692)	73.568 (15.366)	0.554 (0.167)	0.257 (0.155)
		ODS	61.778 (3.832)	44.526 (6.995)	37.222 (3.832)	0.583 (0.050)	0.624 (0.039)
		MMHC	68.218 (3.698)	57.378 (6.753)	30.782 (3.698)	0.544 (0.038)	0.689 (0.037)
		K2mix	72.686 (1.700)	3.240 (2.826)	26.314 (1.700)	0.959 (0.035)	0.734 (0.017)
		PClog	51.812 (5.414)	56.278 (7.107)	47.188 (5.414)	0.480 (0.055)	0.523 (0.055)
	200	PKBIC	34.552 (3.696)	89.626 (8.027)	60.448 (3.696)	0.279 (0.030)	0.364 (0.039)
		Or-PPGM	23.310 (3.416)	107.178 (4.503)	71.690 (3.416)	0.179 (0.024)	0.245 (0.036)
		Or-LPGM	33.604 (5.102)	120.066 (21.818)	61.396 (5.102)	0.221 (0.034)	0.354 (0.054)
		PDN	12.786 (2.974)	103.698 (5.743)	82.214 (2.974)	0.110 (0.027)	0.135 (0.031)

Table B.6 – continued from previous page

Graph	n	Algorithm	TP	FP	FN	PPV	Se
Hub		ODS	7.456 (5.303)	139.400 (23.446)	87.544 (5.303)	0.052 (0.038)	0.078 (0.056)
		MMHC	15.696 (3.269)	93.386 (8.165)	79.304 (3.269)	0.144 (0.031)	0.165 (0.034)
		K2mix	16.492 (3.354)	125.958 (10.452)	78.508 (3.354)	0.116 (0.024)	0.174 (0.035)
		PClog	6.548 (2.275)	38.332 (4.218)	88.452 (2.275)	0.145 (0.046)	0.069 (0.024)
	1000	PKBIC	58.894 (1.247)	39.518 (5.958)	36.106 (1.247)	0.601 (0.037)	0.620 (0.013)
		Or-PPGM	65.716 (2.282)	93.514 (4.991)	29.284 (2.282)	0.413 (0.016)	0.692 (0.024)
		Or-LPGM	67.992 (2.204)	24.026 (6.270)	27.008 (2.204)	0.742 (0.050)	0.716 (0.023)
		PDN	48.904 (2.556)	58.606 (3.902)	46.096 (2.556)	0.455 (0.027)	0.515 (0.027)
		ODS	17.222 (9.947)	77.704 (14.054)	77.778 (9.947)	0.184 (0.109)	0.181 (0.105)
		MMHC	47.918 (5.257)	60.684 (8.779)	47.082 (5.257)	0.443 (0.055)	0.504 (0.055)
		K2mix	46.384 (2.678)	3.524 (1.759)	48.616 (2.678)	0.930 (0.033)	0.488 (0.028)
		PClog	24.278 (3.993)	61.378 (4.362)	70.722 (3.993)	0.283 (0.038)	0.256 (0.042)
	2000	PKBIC	60.594 (0.689)	27.658 (5.061)	34.406 (0.689)	0.689 (0.040)	0.638 (0.007)
		Or-PPGM	72.768 (1.396)	88.504 (4.943)	22.232 (1.396)	0.452 (0.014)	0.766 (0.015)
		Or-LPGM	72.740 (1.257)	5.704 (2.679)	22.260 (1.257)	0.928 (0.031)	0.766 (0.013)
		PDN	56.198 (1.222)	48.538 (2.871)	38.802 (1.222)	0.537 (0.017)	0.592 (0.013)
		ODS	22.248 (10.850)	62.810 (14.200)	72.752 (10.850)	0.266 (0.139)	0.234 (0.114)
		MMHC	57.448 (6.538)	46.008 (8.379)	37.552 (6.538)	0.557 (0.069)	0.605 (0.069)
		K2mix	55.312 (1.498)	0.606 (0.769)	39.688 (1.498)	0.989 (0.013)	0.582 (0.016)
		PClog	30.832 (4.414)	68.980 (4.932)	64.168 (4.414)	0.309 (0.039)	0.325 (0.046)
Random	200	PKBIC	50.116 (3.788)	81.028 (7.412)	58.884 (3.788)	0.383 (0.031)	0.460 (0.035)
		Or-PPGM	41.254 (3.841)	100.784 (4.320)	67.746 (3.841)	0.290 (0.024)	0.378 (0.035)
		Or-LPGM	49.328 (4.380)	115.034 (22.473)	59.672 (4.380)	0.305 (0.040)	0.453 (0.040)
		PDN	17.548 (2.600)	95.862 (5.190)	91.452 (2.600)	0.155 (0.024)	0.161 (0.024)
		ODS	22.630 (4.154)	135.820 (21.734)	86.370 (4.154)	0.145 (0.030)	0.208 (0.038)
		MMHC	22.076 (4.159)	97.612 (7.635)	86.924 (4.159)	0.185 (0.034)	0.203 (0.038)
		K2mix	22.148 (3.518)	119.108 (9.842)	86.852 (3.518)	0.157 (0.026)	0.203 (0.032)
		PClog	21.080 (3.435)	37.376 (5.092)	87.920 (3.435)	0.362 (0.058)	0.193 (0.032)
	1000	PKBIC	81.430 (1.865)	32.738 (4.784)	27.570 (1.865)	0.714 (0.031)	0.747 (0.017)
		Or-PPGM	84.460 (1.964)	90.994 (4.553)	24.540 (1.964)	0.482 (0.014)	0.775 (0.018)
		Or-LPGM	81.102 (2.351)	15.744 (4.862)	27.898 (2.351)	0.840 (0.041)	0.744 (0.022)
		PDN	35.132 (2.159)	70.318 (3.116)	73.868 (2.159)	0.333 (0.020)	0.322 (0.020)
		ODS	42.662 (4.883)	57.110 (7.878)	66.338 (4.883)	0.429 (0.055)	0.391 (0.045)
		MMHC	54.782 (4.536)	75.370 (8.169)	54.218 (4.536)	0.422 (0.040)	0.503 (0.042)
		K2mix	50.454 (2.416)	3.200 (1.653)	58.546 (2.416)	0.941 (0.029)	0.463 (0.022)
		PClog	52.474 (3.260)	52.130 (5.727)	56.526 (3.260)	0.503 (0.037)	0.481 (0.030)
	2000	PKBIC	85.326 (0.922)	21.860 (4.457)	23.674 (0.922)	0.797 (0.033)	0.783 (0.008)
		Or-PPGM	89.416 (0.955)	90.278 (4.744)	19.584 (0.955)	0.498 (0.014)	0.820 (0.009)
		Or-LPGM	87.088 (1.512)	2.482 (1.702)	21.912 (1.512)	0.973 (0.018)	0.799 (0.014)
		PDN	37.664 (1.724)	66.620 (2.574)	71.336 (1.724)	0.361 (0.016)	0.346 (0.016)
		ODS	51.472 (4.955)	40.012 (5.654)	57.528 (4.955)	0.563 (0.056)	0.472 (0.045)
		MMHC	63.616 (4.222)	58.320 (6.914)	45.384 (4.222)	0.523 (0.042)	0.584 (0.039)
		K2mix	61.286 (2.023)	0.778 (0.864)	47.714 (2.023)	0.988 (0.014)	0.562 (0.019)
		PClog	56.950 (3.480)	55.198 (6.063)	52.050 (3.480)	0.509 (0.039)	0.522 (0.032)

B.3 Appendix B.3

Table B.7, and Table B.8 report TP, FP, FN, PPV and Se for each of methods considered in Section 5.2. Two different graph dimensions, i.e., $p = 10, 100$, and three graph structures (see Figure 4.1 and Figure 4.1) are considered.

Table B.7: Simulation results from 500 replicates of the DAGs shown in Figure 4.1 for $p = 10$ variables with Poisson node conditional distribution. Monte Carlo means (standard deviations) are shown for TP, FP, FN, PPV and Se.

Graph	n	Algorithm	TP	FP	FN	PPV	Se
Scale-free	200	plearnDAG	4.000 (1.218)	1.350 (0.978)	5.000 (1.218)	0.749 (0.172)	0.444 (0.135)
		olearnDAG	3.786 (1.122)	1.620 (0.970)	5.214 (1.122)	0.703 (0.163)	0.421 (0.125)
		llearnDAG	4.360 (1.227)	0.752 (0.792)	4.640 (1.227)	0.854 (0.154)	0.484 (0.136)
		PDN	3.600 (0.688)	5.628 (0.680)	5.400 (0.688)	0.390 (0.068)	0.400 (0.076)
		ODS	4.724 (1.100)	10.722 (3.344)	4.276 (1.100)	0.318 (0.096)	0.525 (0.122)
		MMHC	2.988 (1.120)	4.374 (1.309)	6.012 (1.120)	0.407 (0.143)	0.332 (0.124)
		PClog	4.314 (0.868)	2.094 (0.918)	4.686 (0.868)	0.678 (0.114)	0.479 (0.096)
	1000	plearnDAG	6.748 (1.124)	2.048 (1.153)	2.252 (1.124)	0.769 (0.129)	0.750 (0.125)
		olearnDAG	6.220 (1.017)	2.660 (1.007)	2.780 (1.017)	0.701 (0.111)	0.691 (0.113)
		llearnDAG	7.686 (1.050)	0.512 (0.718)	1.314 (1.050)	0.854 (0.117)	0.938 (0.086)
		PDN	3.242 (0.477)	5.904 (0.308)	5.758 (0.477)	0.353 (0.040)	0.360 (0.053)
		ODS	6.274 (1.081)	5.704 (2.017)	2.726 (1.081)	0.534 (0.121)	0.697 (0.120)
		MMHC	4.898 (1.242)	4.294 (1.221)	4.102 (1.242)	0.532 (0.130)	0.544 (0.138)
		PClog	6.544 (0.708)	2.434 (0.836)	2.456 (0.708)	0.731 (0.082)	0.727 (0.079)
	2000	plearnDAG	7.420 (1.231)	1.530 (1.301)	1.580 (1.231)	0.831 (0.142)	0.824 (0.137)
		olearnDAG	6.414 (0.897)	2.644 (0.959)	2.586 (0.897)	0.709 (0.102)	0.713 (0.100)
		llearnDAG	8.386 (0.789)	0.200 (0.439)	0.614 (0.789)	0.977 (0.050)	0.932 (0.088)
		PDN	3.110 (0.320)	5.986 (0.118)	5.890 (0.320)	0.341 (0.023)	0.346 (0.036)
		ODS	6.856 (0.989)	3.922 (1.569)	2.144 (0.989)	0.644 (0.120)	0.762 (0.110)
		MMHC	5.534 (0.855)	3.758 (1.038)	3.466 (0.855)	0.598 (0.098)	0.615 (0.095)
		PClog	6.770 (0.618)	2.356 (0.814)	2.230 (0.618)	0.744 (0.077)	0.752 (0.069)
Hub	200	plearnDAG	1.472 (0.903)	0.186 (0.460)	6.528 (0.903)	0.895 (0.254)	0.184 (0.113)
		olearnDAG	1.120 (0.779)	0.544 (0.614)	6.880 (0.779)	0.687 (0.356)	0.140 (0.097)
		llearnDAG	1.912 (0.765)	0.200 (0.457)	6.088 (0.765)	0.921 (0.178)	0.239 (0.096)
		PDN	1.928 (0.315)	7.548 (0.904)	6.072 (0.315)	0.205 (0.035)	0.241 (0.039)
		ODS	2.560 (1.235)	6.882 (3.167)	5.440 (1.235)	0.291 (0.153)	0.320 (0.154)
		MMHC	1.116 (0.464)	2.780 (1.194)	6.884 (0.464)	0.309 (0.125)	0.140 (0.058)
		PClog	1.644 (0.786)	1.346 (0.734)	6.356 (0.786)	0.541 (0.190)	0.206 (0.098)
	1000	plearnDAG	4.232 (0.839)	0.360 (0.675)	3.768 (0.839)	0.922 (0.146)	0.529 (0.105)
		olearnDAG	3.488 (0.734)	1.104 (0.615)	4.512 (0.734)	0.760 (0.134)	0.436 (0.092)
		llearnDAG	4.120 (0.804)	0.240 (0.463)	3.880 (0.804)	0.948 (0.104)	0.515 (0.101)
		PDN	2.018 (0.133)	7.152 (0.744)	5.982 (0.133)	0.221 (0.020)	0.252 (0.017)
		ODS	3.406 (0.916)	3.354 (1.824)	4.594 (0.916)	0.530 (0.188)	0.426 (0.114)
		MMHC	1.146 (0.661)	3.878 (0.932)	6.854 (0.661)	0.230 (0.113)	0.143 (0.083)
		PClog	3.560 (0.629)	1.302 (0.687)	4.440 (0.629)	0.737 (0.111)	0.445 (0.079)

Table B.7 – continued from previous page

Graph	n	Algorithm	TP	FP	FN	PPV	Se
	2000	plearnDAG	4.608 (0.704)	0.356 (0.692)	3.392 (0.704)	0.929 (0.137)	0.576 (0.088)
		olearnDAG	3.796 (0.599)	1.170 (0.608)	4.204 (0.599)	0.765 (0.117)	0.474 (0.075)
		llearnDAG	4.600 (0.563)	0.170 (0.421)	3.400 (0.563)	0.967 (0.082)	0.575 (0.070)
		PDN	2.004 (0.063)	7.050 (0.823)	5.996 (0.063)	0.223 (0.021)	0.250 (0.008)
		ODS	3.824 (0.794)	1.884 (1.234)	4.176 (0.794)	0.686 (0.181)	0.478 (0.099)
		MMHC	1.074 (0.538)	4.130 (0.752)	6.926 (0.538)	0.207 (0.084)	0.134 (0.067)
		PClog	3.908 (0.390)	1.274 (0.663)	4.092 (0.390)	0.761 (0.093)	0.488 (0.049)
Random	200	plearnDAG	0.776 (0.748)	0.516 (0.612)	7.224 (0.748)	0.589 (0.441)	0.097 (0.093)
		olearnDAG	0.538 (0.708)	0.762 (0.650)	7.462 (0.708)	0.380 (0.433)	0.067 (0.089)
		llearnDAG	1.422 (0.813)	0.302 (0.562)	6.578 (0.813)	0.848 (0.285)	0.178 (0.102)
		PDN	2.406 (0.728)	7.238 (1.023)	5.594 (0.728)	0.250 (0.075)	0.301 (0.091)
		ODS	2.390 (1.249)	7.250 (3.219)	5.610 (1.249)	0.268 (0.156)	0.299 (0.156)
		MMHC	2.454 (0.987)	1.312 (1.135)	5.546 (0.987)	0.689 (0.244)	0.307 (0.123)
		PClog	1.076 (1.195)	1.668 (0.918)	6.924 (1.195)	0.331 (0.361)	0.134 (0.149)
	1000	plearnDAG	2.340 (1.215)	0.834 (0.832)	5.660 (1.215)	0.728 (0.279)	0.292 (0.152)
		olearnDAG	2.022 (1.188)	1.254 (0.817)	5.978 (1.188)	0.591 (0.252)	0.253 (0.149)
		llearnDAG	3.664 (0.960)	0.568 (0.706)	4.336 (0.960)	0.870 (0.156)	0.458 (0.120)
		PDN	2.974 (0.171)	6.292 (0.729)	5.026 (0.171)	0.323 (0.031)	0.372 (0.021)
		ODS	3.402 (1.099)	4.100 (1.827)	4.598 (1.099)	0.469 (0.171)	0.425 (0.137)
		MMHC	4.352 (0.889)	1.096 (1.038)	3.648 (0.889)	0.811 (0.169)	0.544 (0.111)
		PClog	3.860 (1.088)	1.398 (0.597)	4.140 (1.088)	0.723 (0.137)	0.482(0.136)
	2000	plearnDAG	3.230 (1.348)	0.720 (0.784)	4.770 (1.348)	0.802 (0.231)	0.404 (0.168)
		olearnDAG	2.764 (1.361)	1.292 (0.803)	5.236 (1.361)	0.653 (0.232)	0.346 (0.170)
		llearnDAG	4.656 (0.927)	0.456 (0.643)	3.344 (0.927)	0.914 (0.120)	0.582 (0.116)
		PDN	2.994 (0.077)	6.156 (0.664)	5.006 (0.077)	0.329 (0.025)	0.374 (0.010)
		ODS	3.884 (1.132)	2.684 (1.517)	4.116 (1.132)	0.604 (0.196)	0.486 (0.141)
		MMHC	5.012 (0.840)	0.998 (0.932)	2.988 (0.840)	0.840 (0.140)	0.626 (0.105)
		PClog	4.808 (0.590)	1.228 (0.465)	3.192 (0.590)	0.798 (0.066)	0.601 (0.074)

Table B.8: Simulation results from 500 replicates of the DAGs shown in Figure 4.2 for $p = 100$ variables with Poisson node conditional distribution. Monte Carlo means (standard deviations) are shown for TP, FP, FN, PPV and Se.

Graph	n	Algorithm	TP	FP	FN	PPV	Se
	200	plearnDAG	30.320 (3.337)	21.544 (3.841)	68.680 (3.337)	0.586 (0.062)	0.306 (0.034)
		olearnDAG	29.432 (3.423)	22.596 (3.818)	69.568 (3.423)	0.567 (0.062)	0.297 (0.035)
		llearnDAG	44.134 (3.491)	10.052 (3.487)	54.866 (3.491)	0.816 (0.057)	0.446 (0.035)
		PDN	26.572 (5.971)	72.228 (21.678)	72.428 (5.971)	0.299 (0.123)	0.268 (0.060)
		ODS	33.782 (4.472)	152.354 (22.276)	65.218 (4.472)	0.184 (0.032)	0.341 (0.045)
		MMHC	32.828 (4.439)	89.412 (7.336)	66.172 (4.439)	0.269 (0.037)	0.332 (0.045)
		PClog	29.180 (3.413)	39.388 (4.964)	69.820 (3.413)	0.427 (0.051)	0.295 (0.034)

Table B.8 – continued from previous page

Graph	n	Algorithm	TP	FP	FN	PPV	Se
Scale-free	1000	plearnDAG	48.487 (3.300)	39.863 (4.081)	50.513 (3.300)	0.549 (0.039)	0.490 (0.033)
		olearnDAG	47.443 (3.887)	41.173 (4.518)	51.557 (3.887)	0.536 (0.045)	0.479 (0.039)
		llearnDAG	70.140 (2.985)	12.177 (3.920)	28.860 (2.985)	0.853 (0.043)	0.708 (0.030)
		PDN	29.558 (14.034)	39.488 (22.757)	69.442 (14.034)	0.496 (0.160)	0.299 (0.142)
		ODS	53.910 (3.960)	64.874 (9.449)	45.090 (3.960)	0.456 (0.048)	0.545 (0.040)
		MMHC	61.954 (4.035)	69.644 (7.537)	37.046 (4.035)	0.472 (0.037)	0.626 (0.041)
		PClog	48.790 (4.003)	53.670 (6.318)	50.210 (4.003)	0.477 (0.047)	0.493 (0.040)
	2000	plearnDAG	51.382 (3.245)	41.598 (4.143)	47.618 (3.245)	0.553 (0.038)	0.519 (0.033)
		olearnDAG	49.462 (4.163)	44.010 (4.986)	49.538 (4.163)	0.530 (0.047)	0.500 (0.042)
		llearnDAG	75.954 (2.918)	14.522 (3.531)	23.046 (2.918)	0.840 (0.036)	0.767 (0.029)
		PDN	25.432 (15.366)	28.634 (20.692)	73.568 (15.366)	0.554 (0.167)	0.257 (0.155)
		ODS	61.778 (3.832)	44.526 (6.995)	37.222 (3.832)	0.583 (0.050)	0.624 (0.039)
		MMHC	68.218 (3.698)	57.378 (6.753)	30.782 (3.698)	0.544 (0.038)	0.689 (0.037)
		PClog	51.812 (5.414)	56.278 (7.107)	47.188 (5.414)	0.480 (0.055)	0.523 (0.055)
Hub	200	plearnDAG	9.713 (2.570)	7.810 (2.665)	85.287 (2.570)	0.557 (0.117)	0.102 (0.027)
		olearnDAG	9.573 (2.583)	7.647 (2.637)	85.427 (2.583)	0.559 (0.119)	0.101 (0.027)
		llearnDAG	10.827 (2.923)	6.533 (2.386)	84.173 (2.923)	0.625 (0.115)	0.114 (0.031)
		PDN	12.786 (2.974)	103.698 (5.743)	82.214 (2.974)	0.110 (0.027)	0.135 (0.031)
		ODS	7.456 (5.303)	139.400 (23.446)	87.544 (5.303)	0.052 (0.038)	0.078 (0.056)
		MMHC	15.696 (3.269)	93.386 (8.165)	79.304 (3.269)	0.144 (0.031)	0.165 (0.034)
		PClog	6.548 (2.275)	38.332 (4.218)	88.452 (2.275)	0.145 (0.046)	0.069 (0.024)
	1000	plearnDAG	47.487 (3.031)	15.683 (3.520)	47.513 (3.031)	0.753 (0.049)	0.500 (0.032)
		olearnDAG	45.250 (7.329)	17.430 (6.489)	49.750 (7.329)	0.721 (0.110)	0.476 (0.077)
		llearnDAG	47.707 (2.452)	10.920 (2.931)	47.293 (2.452)	0.815 (0.042)	0.502 (0.026)
		PDN	48.904 (2.556)	58.606 (3.902)	46.096 (2.556)	0.455 (0.027)	0.515 (0.027)
		ODS	17.222 (9.947)	77.704 (14.054)	77.778 (9.947)	0.184 (0.109)	0.181 (0.105)
		MMHC	47.918 (5.257)	60.684 (8.779)	47.082 (5.257)	0.443 (0.055)	0.504 (0.055)
		PClog	24.278 (3.993)	61.378 (4.362)	70.722 (3.993)	0.283 (0.038)	0.256 (0.042)
	2000	plearnDAG	53.627 (2.344)	20.957 (3.252)	41.373 (2.344)	0.720 (0.037)	0.564 (0.025)
		olearnDAG	50.123 (8.320)	24.830 (9.141)	44.877 (8.320)	0.670 (0.115)	0.528 (0.088)
		llearnDAG	53.587 (1.878)	13.207 (2.487)	41.413 (1.878)	0.803 (0.031)	0.564 (0.020)
		PDN	56.198 (1.222)	48.538 (2.871)	38.802 (1.222)	0.537 (0.017)	0.592 (0.013)
		ODS	22.248 (10.850)	62.810 (14.200)	72.752 (10.850)	0.266 (0.139)	0.234 (0.114)
		MMHC	57.448 (6.538)	46.008 (8.379)	37.552 (6.538)	0.557 (0.069)	0.605 (0.069)
		PClog	30.832 (4.414)	68.980 (4.932)	64.168 (4.414)	0.309 (0.039)	0.325 (0.046)
Random	200	plearnDAG	16.853 (2.911)	14.240 (2.911)	92.147 (2.911)	0.543 (0.074)	0.155 (0.027)
		olearnDAG	17.027 (3.068)	14.177 (3.113)	91.973 (3.068)	0.547 (0.076)	0.156 (0.028)
		llearnDAG	20.457 (3.190)	9.490 (3.237)	88.543 (3.190)	0.687 (0.086)	0.188 (0.029)
		PDN	17.548 (2.600)	95.862 (5.190)	91.452 (2.600)	0.155 (0.024)	0.161 (0.024)
		ODS	22.630 (4.154)	135.820 (21.734)	86.370 (4.154)	0.145 (0.030)	0.208 (0.038)
		MMHC	22.076 (4.159)	97.612 (7.635)	86.924 (4.159)	0.185 (0.034)	0.203 (0.038)
		PClog	21.080 (3.435)	37.376 (5.092)	87.920 (3.435)	0.362 (0.058)	0.193 (0.032)
	1000	plearnDAG	47.463 (2.559)	32.973 (3.236)	61.537 (2.559)	0.590 (0.032)	0.435 (0.023)
		olearnDAG	49.347 (3.686)	34.103 (4.141)	59.653 (3.686)	0.592 (0.044)	0.453 (0.034)

Table B.8 – continued from previous page

Graph	n	Algorithm	TP	FP	FN	PPV	Se
		llearnDAG	58.483 (3.952)	12.260 (3.460)	50.517 (3.952)	0.827 (0.045)	0.537 (0.036)
		PDN	35.132 (2.159)	70.318 (3.116)	73.868 (2.159)	0.333 (0.020)	0.322 (0.020)
		ODS	42.662 (4.883)	57.110 (7.878)	66.338 (4.883)	0.429 (0.055)	0.391 (0.045)
		MMHC	54.782 (4.536)	75.370 (8.169)	54.218 (4.536)	0.422 (0.040)	0.503 (0.042)
		PClog	52.474 (3.260)	52.130 (5.727)	56.526 (3.260)	0.503 (0.037)	0.481 (0.030)
	2000	plearnDAG	52.530 (2.178)	36.863 (3.076)	56.470 (2.178)	0.588 (0.027)	0.482 (0.020)
		olearnDAG	54.483 (3.461)	38.667 (4.158)	54.517 (3.461)	0.585 (0.040)	0.500 (0.032)
		llearnDAG	68.920 (3.393)	8.980 (2.834)	40.080 (3.393)	0.885 (0.035)	0.632 (0.031)
		PDN	37.664 (1.724)	66.620 (2.574)	71.336 (1.724)	0.361 (0.016)	0.346 (0.016)
		ODS	51.472 (4.955)	40.012 (5.654)	57.528 (4.955)	0.563 (0.056)	0.472 (0.045)
		MMHC	63.616 (4.222)	58.320 (6.914)	45.384 (4.222)	0.523 (0.042)	0.584 (0.039)
		PClog	56.950 (3.480)	55.198 (6.063)	52.050 (3.480)	0.509 (0.039)	0.522 (0.032)

Bibliography

Akaike, H. (1974) A new look at the statistical model identification. *IEEE transactions on automatic control* **19**(6), 716–723.

Aliferis, C. F., Tsamardinos, I. and Statnikov, A. (2003) Hiton: a novel Markov blanket algorithm for optimal variable selection. In *AMIA Annual Symposium Proceedings*, volume 2003, p. 21.

Allen, G. and Liu, Z. (2013) A local Poisson graphical model for inferring networks from sequencing data. *NanoBioscience, IEEE Transactions on* **12**(3), 189–198.

Banerjee, O., Ghaoui, L. E. and dAspremont, A. (2008) Model selection through sparse maximum likelihood estimation. *Journal of Machine Learning Research* **9**(Mar), 485–516.

Besag, J. (1974) Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society. Series B (Methodological)* pp. 192–236.

Breiman, L., Friedman, J. H., Olshen, R. A. and Stone, C. J. (1984) Classification and regression trees Belmont. *CA: Wadsworth International Group* .

Bullard, J. H., Purdom, E., Hansen, K. D. and Dudoit, S. (2010) Evaluation of statistical methods for normalization and differential expression in mRNA-Seq experiments. *BMC Bioinformatics* **11**(1), 94.

Burnham, K. P. and Anderson, D. R. (2003) *Model selection and multimodel inference: a practical information-theoretic approach.* Springer Science & Business Media.

Cai, T., Liu, W. and Luo, X. (2011) A constrained l_1 minimization approach to sparse precision matrix estimation. *Journal of the American Statistical Association* **106**(494), 594–607.

Chickering, D. M. and Meek, C. (2002) Finding optimal Bayesian networks. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pp. 94–102.

Colombo, D. and Maathuis, M. H. (2014) Order-independent constraint-based causal structure learning. *Journal of Machine Learning Research* **15**(1), 3741–3782.

Cooper, G. F. and Herskovits, E. (1992) A bayesian method for the induction of probabilistic networks from data. *Machine learning* **9**(4), 309–347.

Dor, D. and Tarsi, M. (1992) A simple algorithm to construct a consistent extension of a partially oriented graph. *Technicial Report R-185, Cognitive Systems Laboratory, UCLA* .

Fraley, C. and Raftery, A. E. (2002) Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association* **97**(458), 611–631.

Friedman, J., Hastie, T. and Tibshirani, R. (2008) Sparse inverse covariance estimation with the graphical lasso. *Biostatistics* **9**(3), 432–441.

Friedman, J., Hastie, T. and Tibshirani, R. (2009) glmnet: Lasso and elastic-net regularized generalized linear models. *R package version* **1**.

Friedman, J., Hastie, T. and Tibshirani, R. (2010) Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software* **33**(1), 1.

Gallopin, M., Rau, A. and Jaffrézic, F. (2013) A hierarchical Poisson log-normal model for network inference from RNA sequencing data. *PloS One* **8**(10), e77503.

Hadiji, F., Molina, A., Natarajan, S. and Kersting, K. (2015) Poisson dependency networks: Gradient boosted models for multivariate count data. *Machine Learning* **100**(2-3), 477–507.

Haughton, D. M. *et al.* (1988) On the choice of a model to fit data from an exponential family. *The Annals of Statistics* **16**(1), 342–355.

Heckerman, D., Geiger, D. and Chickering, D. M. (1995) Learning Bayesian networks: The combination of knowledge and statistical data. *Machine learning* **20**(3), 197–243.

Kalisch, M. and Bühlmann, P. (2007) Estimating high-dimensional directed acyclic graphs with the PC-algorithm. *Journal of Machine Learning Research* **8**(Mar), 613–636.

Kruskal, W. H. (1958) Ordinal measures of association. *Journal of the American Statistical Association* **53**(284), 814–861.

Lauritzen, S. L. (1996) *Graphical Models*. Volume 17. Clarendon Press, Oxford.

Lauritzen, S. L., Dawid, A. P., Larsen, B. N. and Leimer, H. G. (1990) Independence properties of directed Markov fields. *Networks* **20**(5), 491–505.

Lehmann, E. L. (1986) *Testing statistical hypotheses (2nd ed).* New York, NY: Wiley.

Li, J., Witten, D. M., Johnstone, I. M. and Tibshirani, R. (2012) Normalization, testing, and false discovery rate estimation for RNA-sequencing data. *Biostatistics* **13**(3), 523–538.

Liu, H., Han, F., Yuan, M., Lafferty, J. and Wasserman, L. (2012) The nonparanormal skeptic. *arXiv preprint arXiv:1206.6488* .

Liu, H., Lafferty, J. and Wasserman, L. (2009) The nonparanormal: Semiparametric estimation of high dimensional undirected graphs. *Journal of Machine Learning Research* **10**(Oct), 2295–2328.

Liu, H., Roeder, K. and Wasserman, L. (2010) Stability approach to regularization selection (stars) for high dimensional graphical models. In *Advances in neural information processing systems*, pp. 1432–1440.

Meek, C. (1995) Strong completeness and faithfulness in Bayesian networks. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pp. 411–418.

Meinshausen, N. and Bühlmann, P. (2006) High-dimensional graphs and variable selection with the lasso. *The Annals of Statistics* pp. 1436–1462.

Park, G. and Raskutti, G. (2015) Learning large-scale Poisson DAG models based on overdispersion scoring. In *Advances in Neural Information Processing Systems*, pp. 631–639.

Pearl, J. and Paz, A. (1985) *Graphoids: A graph-based logic for reasoning about relevance relations.* University of California (Los Angeles). Computer Science Department.

Peña, J. M., Nilsson, R., Björkegren, J. and Tegnér, J. (2009) An algorithm for reading dependencies from the minimal undirected independence map of a graphoid that satisfies weak transitivity. *Journal of Machine Learning Research* **10**(May), 1071–1094.

Peters, J. and Bühlmann, P. (2013) Identifiability of Gaussian structural equation models with equal error variances. *Biometrika* **101**(1), 219–228.

Peters, J., Mooij, J., Janzing, D. and Schölkopf, B. (2012) Identifiability of causal graphs using functional models. *arXiv preprint arXiv:1202.3757* .

Sadeghi, K. (2017) Faithfulness of probability distributions and graphs. *arXiv preprint arXiv:1701.08366* .

Schelldorfer, J., Meier, L. and Bühlmann, P. (2014) Glmmlasso: an algorithm for high-dimensional generalized linear mixed models using l1-penalization. *Journal of Computational and Graphical Statistics* **23**(2), 460–477.

Schwarz, G. (1978) Estimating the dimension of a model. *The Annals of Statistics* **6**(2), 461–464.

Shimizu, S., Hoyer, P. O., Hyvärinen, A. and Kerminen, A. (2006) A linear non-Gaussian acyclic model for causal discovery. *Journal of Machine Learning Research* **7**(Oct), 2003–2030.

Spirtes, P., Glymour, C. N. and Scheines, R. (2000) *Causation, prediction, and search*. MIT press.

Tsamardinos, I., Brown, L. E. and Aliferis, C. F. (2006) The max-min hill-climbing Bayesian network structure learning algorithm. *Machine learning* **65**(1), 31–78.

Verzelen, N. and Villers, F. (2009) Tests for Gaussian graphical models. *Computational Statistics & Data Analysis* **53**(5), 1894–1905.

Volinia, S., Galasso, M., Sana, M. E., Wise, T. F., Palatini, J., Huebner, K. and Croce, C. M. (2012) Breast cancer signatures for invasiveness and prognosis defined by deep sequencing of microRNA. *Proceedings of the National Academy of Sciences* **109**(8), 3024–3029.

Wagenmakers, E. J. and Farrell, S. (2004) AIC model selection using Akaike weights. *Psychonomic Bulletin & Review* **11**(1), 192–196.

Yan, X., Chen, X., Liang, H., Deng, T., Chen, W., Zhang, S., Liu, M., Gao, X., Liu, Y., Zhao, C. *et al.* (2014) miR-143 and miR-145 synergistically regulate ERBB3 to suppress cell proliferation and invasion in breast cancer. *Molecular Cancer* **13**(1), 220.

Yang, E., Allen, G., Liu, Z. and Ravikumar, P. K. (2012) Graphical models via generalized linear models. In *Advances in Neural Information Processing Systems*, pp. 1358–1366.

Yang, E., Ravikumar, P., Allen, G. and Liu, Z. (2013) On Poisson graphical models. In *Advances in Neural Information Processing Systems*, pp. 1718–1726.

Yang, E., Ravikumar, P., Allen, G. I. and Liu, Z. (2015) Graphical models via univariate exponential family distributions. *Journal of Machine Learning Research* **16**(1), 3813–3847.

Yang, Z., Ning, Y. and Liu, H. (2014) On semiparametric exponential family graphical models. *arXiv preprint arXiv:1412.8697* .

Yuan, M. (2010) High dimensional inverse covariance matrix estimation via linear programming. *Journal of Machine Learning Research* **11**(Aug), 2261–2286.

Zhang, L. and Mallick, B. K. (2013) Inferring gene networks from discrete expression data. *Biostatistics* p. kxt021.

Zhang, N., Wang, X., Huo, Q., Sun, M., Cai, C., Liu, Z., Hu, G. and Yang, Q. (2014) MicroRNA-30a suppresses breast tumor growth and metastasis by targeting metadherin. *Oncogene* **33**(24), 3119–3128.

Žitnik, M. and Zupan, B. (2015) Gene network inference by fusing data from diverse distributions. *Bioinformatics* **31**(12), i230–i239.

Thi Kim Hue NGUYEN

CURRICULUM VITAE

Contact Information

University of Padova
Department of Statistics
via Cesare Battisti, 241-243
35121 Padova. Italy.

Tel. +39 049 827 4174
e-mail: nguyen@stat.unipd.it

Current Position

Since November 2014; (expected completion: October 2017)
PhD Student in Statistics, University of Padova.
Thesis title: "Structure Learning of Graphs for Count Data"
Supervisor: Prof. Monica Chiogna

Research interests

- Biostatistics
- Graphical models

Education

September 2013- June 2014
Master 2 degree in Probability and Statistics.
Paul Sabatier University, Faculty of Mathematics, France
Title of dissertation: "A Model for Structured Graph Correlation Inference"
Supervisor: Prof. Jean Michel Loubes

September 2013- June 2014
Master 1 degree in Mathematics.
Vietnam Institute of Mathematics, Vietnam

September 2008- June 2012
Bachelor degree in Mathematics.
Hanoi National University of Education, Faculty of Mathematics, Vietnam
Title of dissertation: "Some generalizations of fixed point theory"
Supervisor: Prof. Thi Thanh Ha NGUYEN

Further education

July 2016
Summer School from gene expression to genomic Network
SPS LabEx

August 2017
Summer School on Graphical Models
Technical University of Denmark

August 2017– September 2017
Data Science Summer School
Ecole Polytechnique

Awards and Scholarship

2014-2017
Cariparo PhD Scholarhip for foreign students, University of Padova, Italy.

2013
Scholarship of Centre international de Mathématiqués et Informatique de Toulouse for Master 2, France.

2012
Scholarship of Vietnam Institute of Mathematics for Master 1, Vietnam.

2012
Award for excellent students from University president, Hanoi University of Education, Vietnam.

2008-2012
Academic Scholarships for excellent students, Hanoi University of Education, Vietnam.

Computer skills

- Matlab, R

Language skills

Vietnamese native; English fluent; French moderate.

Publications

Articles in journals

NGUYEN, T.K.H., Chiogna, M., (2017). Structure Learning of Undirected Graphical Models for Count Data. (In preparation)

Conference presentations

NGUYEN, T.K.H., Chiogna, M. (2016). Learning High Dimensional Directed Acyclic Graphs for Count Data. (poster) *Summer School from gene expression to genomic Network*, Paris, France, 17-22/July.

NGUYEN, T.K.H., Chiogna, M. (2017). Structure Learning of Undirected Graphical Models for Count Data. (poster) *Summer School on Graphical Models*, Tjaro, Sweden, 14-18/August.

NGUYEN, T.K.H., Chiogna, M. (2017). Structure Learning of Undirected Graphical Models for Count Data. (poster) *Data Sciences Summer School*, Paris, France, 28/August- 1/September.

Teaching experience

February 2012 - April 2012
Tutor for the course: "Mathematics"
Hung Vuong High School for Gifted Students, Vietnam.
Instructor: Prof. Thi Hong NGUYEN

February 2011 - March 2011
Tutor for the course: "Mathematics"
Hung Vuong High School for Gifted Students, Vietnam.
Instructor: Prof. Thi Tam NGUYEN

References

Prof. Prof. Monica Chiogna
Department of Statistical Sciences
University of Padova
via Cesare Battisti, 241-243
35121 Padova, Italy.
e-mail: monica@stat.unipd.it

Prof. Jean Michel Loubes
Insitut de Mathématiques de Toulouse
Université Toulouse III- Paul Sabatier
118 Route de Narbonne
31062 Toulouse, France.
e-mail: loubes@math.univ-toulouse.fr