l'université nantes angers le mans
PÔLE DE RECHERCHE ET D'ENSEIGNEMENT SUPÉRIEUR

UNIVERSITÉ DE NANTES

Thèse de Doctorat

Mouna BEN ISHAK

Mémoire présenté en vue de l'obtention du
grade de Docteur de l'Université de Nantes
Docteur de l'Université de Tunis
sous le label de l'Université de Nantes Angers Le Mans

École doctorale : Sciences et technologies de l'information, et mathématiques

Discipline : Informatique, section CNU 27
Unité de recherche : Laboratoire d'informatique de Nantes-Atlantique (LINA)
Laboratoire de recherche opérationnelle, de décision et de contrôle de processus (LARODEC)

Soutenue le
Thèse n° : ED 503

Probabilistic relational models: learning and evaluation

The relational Bayesian networks case

JURY

Président :	**M. Zied ELOUEDI**, Professeur des Universités, Université de Tunis, Tunisie
Rapporteurs :	**M^me^ Céline ROUVEIROL**, Professeur des Universités, Université Paris-Nord, France
	M. Jérôme DARMONT, Professeur des Universités, Université Lumière Lyon 2, France
Examinateur :	**M. Pierre-Henri WUILLEMIN**, Maître de conférence, Université Pierre et Marie Curie, Paris, France
Directeurs de thèse :	**M. Philippe LERAY**, Professeur des Universités, Université de Nantes, France
	M^me^ Nahla BEN AMOR, Professeur des Universités, Université de Tunis, Tunisie

To my father's soul,
to my dear mum,
to my dear brothers,
to all my family members,
to all those who love me.

Acknowledgements

First, I thank the jury's members who honored me by judging my dissertation. I am grateful for the time they spent to do this.

Then, I want to express my deep gratitude and sincere appreciation to my two supervisors: Pr. Nahla Ben Amor who often gave me the courage to move forward in my research, I am thankful for her continuous support, her precious advice and her invaluable help, Pr. Philippe Leray for the constructive discussions I had with him. I was fortunate to benefit from his expertise in this interesting research area and to learn from him. Without them this dissertation would never have been achieved.

I would like to offer my heartiest thanks to my beloved parents for their understanding, support and confidence to which I am forever indebted. Dad, may god bless you, I wish you were here to share with us these moments of joy and success. Mum, you are so supreme, you have always looked out for my future and supported my dreams!

An honorable mention goes to my brothers for listening to my complaints and frustrations and for their continuous help. Words are not enough to express how much I respect and I love them and I am sure that having brought this work forward is the best reward for them.

I would like to express my deepest thanks to my lovely fiancé for his steadfast support, patience and encouragement.

I am thankful to my research colleagues either in LARODEC or in DUKe team for their help, support and encouragement whenever I was in need. I am also thankful to my work colleagues at IHEC-Tunisia for their care and precious friendship. I thank them for the happy time we spent together during my teaching experience at IHEC. I will miss you very much!!!

My greatest gratitude goes to my precious friends who were always here to help me to stay sane through the difficult moments of my thesis.

Let all those who helped me by their support and encouragement find here the expression of my sincere gratitude.

Résumé

Les modèles graphiques probabilistes (MGPs) sont des outils puissants de représentation et de raisonnement dans l'incertain. Les réseaux bayésiens sont des modèles graphiques probabilistes dirigés acycliques conçus par Judea Pearl permettant de coder et de manipuler des distributions de probabilité sur des espaces à grande échelle. Ils ont fait preuve de leur efficacité dans différentes applications du monde réel où l'incertitude est presque un aspect incontournable. Pour se servir de ses modèles il faut avant tout les construire. La structure et les paramètres doivent être déjà définis, sinon, ils doivent être appris à partir de données. Apprendre les paramètres d'un RB consiste à fournir des distributions de probabilités conditionnelles de chaque variable dans le contexte de ses parents dans le graph. Ceci pourrait être réalisé soit en utilisant l'approche statistique du maximum de vraisemblance ou les méthodes d'estimation Bayésiennes. Apprendre la structure d'un RB consiste à fournir une structure du réseau qui traduit les données observées. Cette tâche s'avère plus compliquée surtout que l'espace de recherche est généralement très volumineux et devient rapidement infinie même avec un petit nombre de variables. Plusieurs travaux de recherche ont été menés dans ce sens. Ces travaux sont répartis en trois grandes familles. Les algorithmes à base de contraintes traitent cette question comme étant un problème de satisfaction de contraintes. Ils recherchent des indépendances (dépendances) dans les données, en utilisant des tests statistiques Ensuite, ils essayent de trouver la structure graphique la plus appropriée. Les algorithmes à base de score traitent l'apprentissage de la structure comme étant un problème d'optimisation. Ils évaluent la façon dont la structure correspond aux données en utilisant une fonction de score, le but étant de maximiser cette fonction. Les algorithmes hybrides présentent un mélange des deux premières familles. Ces algorithmes ont présenté de meilleurs résultats en terme de complexité et de scalabilité. Le processus d'évaluation d'un algorithme d'apprentissage de la structure nécessite l'existence d'un réseau théorique à partir duquel on peut faire l'échantillonnage. L'apprentissage est réalisée en utilisant les données générées et le réseau appris est ensuite comparé au réseau théorique en utilisant des mesures d'évaluation. Habituellement, des RBs connus dans la littérature sont utilisés comme réseaux théoriques (e.g., ASIA, INSURANCE). Pour fournir des études comparatives plus rigoureuses, certains chercheurs utilisent un processus de génération aléatoire de RBs synthétiques.

Les RBs ont été développés pour manipuler des données sous forme matricielle simple. Toutefois, et en raison de l'évolution des technologies de communication et de stockage, les données des applications réelles ne respectent plus cette forme. Elles présentent un très grand nombre de dimensions, avec différents types d'entités. Au début des années 2000, un grand intérêt a été adressé au traitement directe des données relationnelles. L'apprentissage relationnel statistique (ARS) est un nouveau domaine de l'apprentissage automatique qui permet de raisonner dans ce contexte. Les réseaux Bayésiens relationnels (RBRs) présentent une extension des réseaux Bayésiens qui permet la production et la manipulation de représentations structurées des données, impliquant des objets décrits par des attributs et participant à des relations, des actions et des événements. La spécification du modèle probabiliste concerne des classes d'objets plutôt que de simples variables aléatoires. Ils ont fait leur preuve dans plusieurs domaines d'application (e.g., l'industrie, l'analyse de la qualité du système, la classification des pages web) et jusqu'à maintenant les travaux sont en cours afin de consolider la spécification théorique de ces modèles et de développer des algorithmes appropriés permettant leur construction de à partir de données relationnelles. L'apprentissage des RBRs est inspiré des méthodes d'apprentissage des RBs standards. Naturellement, leur évaluation est similaire à celle des RBs. Elle est réalisée en utilisant soit des réseaux connus soit des réseaux synthétique générés

aléatoirement. Cependant, ni la première démarche ni la seconde ne sont disponibles dans le contexte relationnel. Même si une panoplie de travaux ont porté, séparément, sur la génération de réseaux Bayésiens et de bases de données relationnelles, aucun travail regroupant ces deux disciplines n'a été recensé. Le manque de RBRs connus dans la littérature ainsi que le manque de processus de génération aléatoire de ces modèles pourraient expliquer le nombre réduit de travaux portant sur leur apprentissage. En effet, malgré la panoplie des travaux existant pour l'apprentissage de la structure des RB, seulement quelques travaux ont été recensés pour l'apprentissage de la structure de leur extension relationnelle. Ces travaux sont des extensions de quelques algorithmes à base de contraintes ou à base de score pour l'apprentissage des RBs. Aucun algorithme hybride n'a été présenté pour les RBRs.

Dans ce travail de thèse, nous proposons deux contributions majeures. Premièrement, Une approche de génération de RBRs allant de la génération du schéma relationnel, de la structure de dépendance et des tables de probabilités à l'instanciation de ce modèle et la population d'une base de données relationnelle. Les données sont générées non seulement à partir de dépendances fonctionnelles mais aussi de dépendances probabilistes. Ce processus s'avère bénéfique pour les chercheurs intéressés par les modèles graphiques relationnels, notamment leur apprentissage à partir de données, afin de comparer leurs travaux dans un cadre commun. Nous discutons aussi de l'adaptation des mesures d'évaluation des algorithmes d'apprentissage de RBs dans le contexte relationnel et nous proposons de nouvelles mesures d'évaluation. Deuxièmement, Une approche hybride pour l'apprentissage de la structure des RBRs à partir d'une instance complète d'une base de données relationnelle. Cette approche présente une extension de l'algorithme MMHC dans le contexte relationnel que nous appelons MMHC relationnel (RMMHC). Cet algorithme est constitué d'une phase de recherche locale assurée par l'algorithme max-min parents and children relationnel (RMMPC), et d'une phase de recherche globale assurée par l'algorithme relationnel de la recherche gloutonne (RGS). Nous menons une étude expérimentale permettant de comparer ce nouvel algorithme d'apprentissage de structure avec les approches déjà existantes. Cette étude se base sur notre processus de génération aléatoire de RBRs et utilise les mesures d'évaluation que nous proposons pour évaluer les résultats d'apprentissage.

Abstract

Probabilistic graphical models offer a framework including famous statistical formalisms for defining complex probability models such as Bayesian networks (BNs). These latter are directed acyclic graphs conceived by Judea Pearl in order to efficiently encode and manipulate probability distributions over high-dimensional spaces. BNs become quickly an important tool to address real world applications where uncertainty is almost an inescapable aspect. To perform probabilistic inference, the BN structure and parameters have to be already defined, if not, they have to be learned from data. Learning BN parameters consists on providing the conditional probability distributions of each variable given its parents in the graph. It can be done using maximum likelihood or Bayesian estimation methods. Learning the BN structure consists on providing a network structure which fits the best way to the observed data. This is a tremendous issue as the search space is usually very large and becomes quickly infinite even with few number of variables. A wealth of literature has been produced that seeks to understand and provide methods of learning structure from data. BN structure learning methods are divided into three main families. The first tackles this issue as a constraint satisfaction problem. Constraint-based algorithms look for independencies (dependencies) in the data, using statistical tests then, try to find the most suitable graphical structure with this information. The second treats learning as an optimization problem. They evaluate how well the structure fits to the data using a score function. So, these Score-based algorithms search for the structure that maximizes this function. The third presents hybrid approaches that mix both of the first two ones. Hybrid approaches perform better than other techniques based only on conditional independence tests or on score functions on the level of scalability and complexity of algorithms. The evaluation process of a BN structure learning algorithm requires the existence of a theoretical (also called gold) network from which one can sample a training data. Then learning is performed using this sampled data and the learned network is compared to the theoretical one using some evaluation metrics. Usually, famous BNs are used in the literature as gold nets (e.g., ASIA, INSURANCE). To provide more rigorous comparative studies, some researchers use a random generation process of synthetic BNs.

BNs have been developed for data in the traditional matrix form. However, due to the development of communication and storage technologies, data management practices have taken further aspects. The data can present a very large number of dimensions, with several different types of entities. In the early 2000s, there has been growing interest in extracting patterns from relational data representation. Statistical relational learning (SRL) is an emerging area of machine learning that enable effective and robust reasoning about relational data structures. Relational Bayesian networks (RBNs) are an extension of Bayesian networks which allow to work with relational database representation rather than propositional data representation. RBNs are interested in producing and manipulating structured representations of the data, involving objects described by attributes and participating in relationships, actions and events. The probability model specification concerns classes of objects rather than simple attributes. They have demonstrate their applicability in several areas (e.g., industry, system quality analysis, web page classification) and till now works are in progress in order to provide a solid theoretical foundation of them and develop appropriate algorithms allowing their construction from relational data. RBNs learning is inspired from standard BNs learning approaches and naturally their evaluation will be similar to BNs evaluation techniques. It is generally done using either real known networks or randomly generate ones. However, neither the first nor the second are available. Even though a panoply of works have focused, separately, on Bayesian Networks and

relational databases random generation, no work has been identified for RBNs on that track. The lack of famous RBNs as well as random process generation may argued the low number of works dealing with RBNs structure learning. In fact, despite the panoply of works presented for standard BNs structure learning, only few works have been devoted to learn their relational extension. Proposed approaches are extensions of some constraint-based and score-based algorithms but no work has been proposed for relational hybrid approach.

This thesis presents two major contributions. First, we propose an algorithmic approach allowing to generate random RBNs from scratch, then populate a database instance. The originality of this process is that it allows to generate synthetic relational data from a randomly generated relational schema and a random set of probabilistic dependencies. Such a process is imperative for statistical relational learning researchers to evaluate the effectiveness of their learning approaches in a common framework. Also, we discuss the adaptation of the evaluation metrics of BNs structure learning algorithms to the relational context and we propose new relational evaluation measurements. Second, we present a new hybrid approach to learn RBNs structure from a complete relational database instance. We develop an extension of the max-min-hill-climbing (MMHC) algorithm that we refer to as relational max-min-hill-climbing (RMMHC). The approach consists of a local search phase ensured by the relational max-min parents and children algorithm (RMMPC), and a global search phase ensured by the relational greedy search algorithm (RGS). We present an experimental study to compare this new learning algorithm with the state-of-the-art approaches. This study is based on our RBNs random generation process and uses the structure learning evaluation metrics that we propose.

Contents

List of Tables

List of Figures

List of Algorithms

Acronyms

A

AGG
Abstract Ground Graph. 51, 58, 80

AIC
Akaike Information Criterion. 17

ANSI
American National Standards Institute. 34

B

BD
Bayesian Dirichlet. 17

BIC
Bayesian Information Criterion. 17

BN
Bayesian Network. 1, 2, 7, 12, 27, 51, 55, 61, 69, 79, 83, 85, 101, 103, 108, 133, 140

C

CBN
Causal Bayesian Network. 2, 28

CF
Collaborative Filtering. 140, 141

CPC
Candidate Parents and Children. 19, 85, 87, 93, 94

CPD
Conditional Probability Distribution. 7, 9, 12, 43, 45, 71, 73, 106

CPDAG
Completed Partially Directed Acycle Graph. 14, 80, 133

D

DAG
Directed Acyclic Graph. 1, 9, 12, 69, 71, 73, 74, 86, 132

DAPER
Directed Acyclic Probabilistic Entity Relational. 47, 57

DBA
Database Administrator. 31

DBMS
Database Management System. 31, 136

DBN
Dynamic Bayesian Network. 28

E

ER
Entity-Relationship. 35, 37, 38

G

GBN
Ground Bayesian Network. 45, 61, 73

GES
Greedy Equivalence Search. 17

GS
Greedy Search. 17, 18

H

HBN
Hierarchical Bayesian Network. 28

I

IC
Inductive Causation. 16

ILP
Inductive Logic Programming. 1

ISO
International Organization for Standardization. 34

J

JPD
Join Probability Distribution. 13

K

KDD
Knowledge Discovery in Databases. 1

M

MBOR
Markov Boundary search using the OR condition. 21, 58

MDL
Minimum Description Length. 17

MMHC
Max-Min Hill climbing. 19, 83, 85, 93, 94, 132

MMPC
Max-Min Parents and Children. 19, 85, 88, 93, 94

O

OOBN
Object-oriented Bayesian Network. 2

P

PC
Peter and Clark. 16, 57, 58

PGM
Probabilistic Graphical Model. 1, 2, 7, 105

PRM
Probabilistic Relational Model. 41

R

RBN
Relational Bayesian Network. 1, 2, 28, 45, 60, 61, 63, 67, 69, 71, 73, 77, 82, 83, 85, 101, 104, 106, 113, 133, 135, 136, 139, 140, 141

RCD
Relational Causal Discovery. 58, 95, 113

RDBMS
Relational Database Management System. 29, 39, 103

RGS
Relational Greedy Search. 93, 95, 113

RMMHC
Relational Max-Min Hill climbing. 83, 85, 94, 95, 113, 135

RMMPC
Relational Max-Min Parents and Children. 85, 95, 113, 135

RPC
Relational Peter and Clark. 57, 113

RS
Recommender Sysytem. 139

RSHD
Relational Structural Hamming Distance. 69, 80, 115

S

SC
Sparse Candidate. 19

SGS
Spirte, Glymour and Scheines. 16

SHD
Structural Hamming Distance. 25, 67, 80, 85

SQL
Structured Query Language. 34, 35

SRL
Statistical Relational Learning. 1, 41

T

TPC

Transaction Processing Performance Council. 38

X

XML

Extensible Markup Language. 104, 105

Introduction

Context

Knowledge discovery in databases (KDD) is concerned with the development of computational theories and tools to assist humans in extracting hidden knowledge from voluminous data. Knowledge extraction is considered as the end product of a data-driven discovery where data mining is considered as a central step. This latter relies on several research areas, notably, statistics, databases and machine learning. Machine learning techniques allow to learn patterns from training data in order to produce models that can be used for prediction and decision making. Several basic machine learning methods have been involved (e.g., decision trees, association rules, Bayesian networks, support vector machines). These methods, known as propositional learning approaches, focus on the propositional or attribute-value representation: a single table of data where rows represent observations, and columns represent variables. However, due to the development of communication and storage technologies, data about the real world is seldom of this form. The data can present a very large number of dimensions, with several different types of entities.

Interests are, now, bigger in extracting patterns from such data representation. Statistical relational learning (SRL) is an emerging area of machine learning that enable to represent, reason, and learn in domains with complex relational and rich probabilistic structure (Heckerman et al., 2007). Other terms have been used in the same context, including probabilistic logic learning and relational data mining. Utmost SRL algorithms come form the field of inductive logic programming (ILP) (Lavrac and Dzeroski, 1993; Muggleton and De Raedt, 1994) considered at the intersection of machine learning and logic programming. Two major directions have been developed in order to use the old machine learning techniques together with relational data representation. The first aim to transform an ILP problem into a propositional form and then uses attribute-value approaches to solve the problem. De Raedt (De Raedt, 1998) treats the relation between attribute-value learning and ILP in detail, showing that propositionalization of some more complex ILP problems is possible in principle, but results in attribute-value problems that are exponentially large (i.e., with non allowable size).The second opts for the development of new data mining techniques that use directly the relational representation of data. Typically these methods are inspired from traditional data mining methods, while trying to fit them to the relational context. Probabilistic graphical models (PGMs) are quite involved in KDD and three main groups of PGMs have been well studied, namely Bayesian networks (BNs) representing Directed Acyclic Graphs (DAGs) (Pearl, 1988), Markov networks representing undirected graphs (Pearl, 1988) and Dependency networks representing bi-directed graphs (Heckerman et al., 2001). Three extensions match to these groups in the relational context that are respectively probabilistic relational models (Koller and Pfeffer, 1998; Pfeffer, 2000), relational Markov networks (Taskar et al.,) and relational dependency networks (Neville and Jensen, 2007). Neville and Jensen (Neville and Jensen, 2007) use the term relational Bayesian networks (RBNs) to refer to Bayesian networks that have been extended to model relational databases (Koller and Pfeffer, 1998; Pfeffer, 2000) and use the term probabilistic relational models in its more general sense to distinguish the family of PGMs that are interested in extracting statistical patterns from relational models. In this thesis, we adopt the terminology proposed by Neville and Jensen, which has not to be confused with the relational Bayesian networks of Jaeger (Jaeger, 1997). These latter are extension of BNs using the first-Order logic whereas RBNs (Koller and Pfeffer, 1998; Pfeffer, 2000) represent a relational extension of BNs, where the probability model specification concerns classes

of objects rather than simple attributes.

BNs are directed acyclic graphs initiated by Judea Pearl in order to efficiently encode and manipulate probability distributions over high-dimensional spaces (Pearl, 1988). BNs become quickly an important tool to address real world applications where uncertainty is almost an inescapable aspect. In 2011, the Turing Award was awarded to J. Pearl for the development of these models which reflect their important role for the industry as well as the scientific community. Several extensions have been proposed in the literature in order to broaden their range of application. We can, for instance, mention *dynamic Bayesian networks* (Murphy, 2002) that enable to model a system whose state evolves over time, *hierarchical Bayesian networks* (Gyftodimos and Flach, 2002) that gives additional knowledge about variables structure by extending the classical formalism of Bayesian networks with composite nodes allowing the aggregation of simpler types. Pearl (Pearl, 2000) focuses on the semantics of intervention and its relation to causality through the *causal Bayesian networks* (CBN) framework.

RBNs belong to another range of extensions defined by their representation paradigm. The object-oriented, entity-relationship and first order logic paradigms were mainly used. *Object-oriented Bayesian networks* (OOBNs) (Koller and Pfeffer, 1997; Pfeffer, 2000; Bangsø and Wuillemin, 2000) are extensions based on the object-oriented paradigm. The OOBNs as defined by (Pfeffer, 2000) have been extended, on their part, using the entity-relationship paradigm by (Koller and Pfeffer, 1998; Pfeffer, 2000) through the relational Bayesian networks, that we focus on in this thesis work. Other entity-relationship extensions have been also proposed (Wellman et al., 1992; Buntine, 1994; Heckerman et al., 2004) yet RBNs remains the most developed one as they also proposed to manage the structural uncertainty feature (Getoor et al., 2007). Also, many other extensions have been proposed based on the first-Order logic (Jaeger, 1997; Laskey, 2008).

Today, most databases have relational capabilities and probabilistic relational models are powerful data mining tools to deal with this data representation, however, it is not obvious to obtain them from domain experts. So, as for classical PGMs, methods to learn them from relational data have to be developed. Especially structure learning remains the most challenging issue, as it is considered as a NP-Hard problem (Koller and Friedman, 2009). It consists on providing the probabilistic structure for a given relational schema and a relational observational dataset that instantiates this schema. Some preliminary work has been recorded for directed relational models (Friedman et al., 1999a; Maier et al., 2010; Maier et al., 2013a). Yet, the manner how these approaches have been evaluated is open to criticisms and to several improvements. The evaluation of the learning approaches is generally done using randomly generated data coming from either real known networks or randomly generated ones. However, neither the first nor the second are available for RBNs. Furthermore, the proposed approaches are adaptations of either score-based or constraint-based approaches to learn BNs structure, however, it has been shown that hybrid approaches provide better results (Tsamardinos et al., 2006) using several benchmarks and metrics (execution time, SHD measure, etc.)

This thesis addresses three main issues:

- How to randomly generate relational Bayesian networks?
- How to compare two RBN structures?
- How to learn RBNs structure from a complete relational observational dataset using an hybrid approach?

Consequently, our first contribution is the proposition of an algorithmic approach allowing to generate random RBNs from scratch, then populate a database instance. The originality of this process is that it allows to generate synthetic relational data from a randomly generated relational schema and a random set of probabilistic dependencies. Second, we propose a new distance-based measure allowing to compare two RBN structures, one with respect to the other. Finally, we provide a new hybrid approach to learn RBNs structure. We prove the effectiveness of our proposal by comparing it with already existing RBNs learning approaches, in a common framework, using our random generation process.

Overview

Figure 1 presents the thesis organization and interdependencies between chapters.

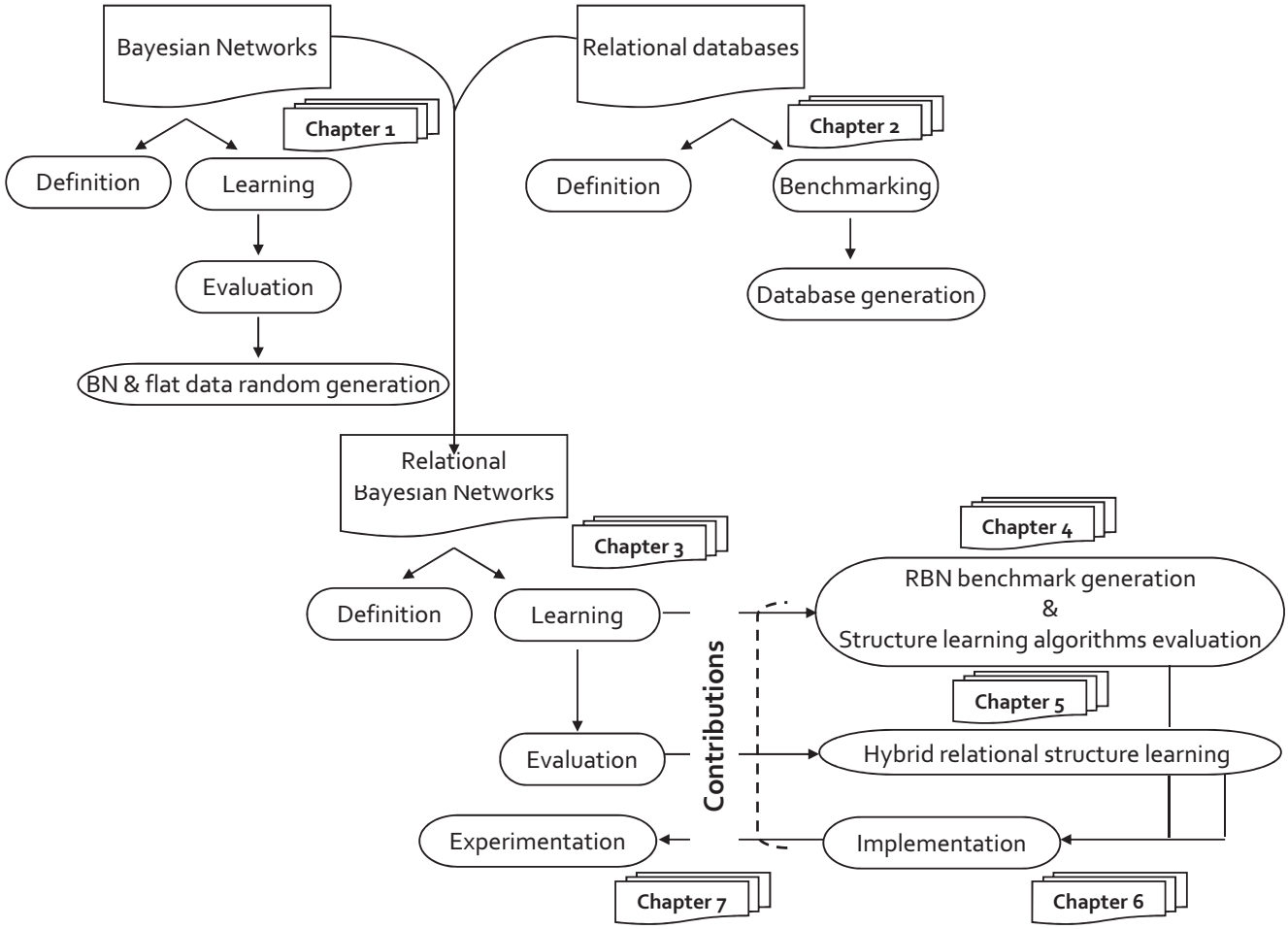


Figure 1: Thesis synopsis

Chapter 1 is dedicated to introduce Bayesian Networks and to make a survey on existing learning as well as evaluating approaches.

Chapter 2 gives an overview of database theory principles. Then, gives a formal definition of the relational model and discusses relational database benchmarking and methods to randomly generate databases.

Chapter 3 introduces relational Bayesian networks, presents their learning approaches and discusses evaluation techniques of RBNs learning approaches.

Chapter 4 provides, firstly, an algorithmic approach allowing to generate random RBNs from scratch to cover the absence of generation processes. The proposed method allows to generate RBNs as well as synthetic relational data from a randomly generated relational schema and a random set of probabilistic dependencies. Then, we present a new distance-based metric allowing to compare an RBN structure with respect to another one.

Chapter 5 provides a new hybrid approach to learn RBNs structure from a complete relational observational dataset. It is an adaptation of the Max-Min Hill Climbing algorithm, described in Chapter 1, to the relational context, that we refer to as Relational Max-Min Hill Climbing algorithm.

Chapter 6 presents environments and softwares used during the development phase. Then, it discusses the chosen policies to develop the generation approach.

Chapter 7 describes the detailed experiment protocol and its execution. It reports experimental results concerning our learning approach compared with state-of-the-art methods.

Publications

This research work on probabilistic graphical models was the subject of the following publications:

- Our preliminary work about OOBNs, not described in this thesis, has been published in (Ben Ishak et al., 2011a) and (Ben Ishak et al., 2011b).
- Our first contribution presented in Chapter 4 has been published in (Ben Ishak et al., 2014c) and (Ben Ishak et al., 2014b). Also, extend version of this work has been accepted for publication in the IDA international journal (Ben Ishak et al.,).
- Our state-of-the-art about RBNs and their application to recommender systems has been published in (Ben Ishak et al., 2013) and (Ben Ishak et al., 2014a).

State-of-the-art

1

Bayesian Networks

Probabilistic graphical models (PGMs) (Koller and Friedman, 2009) offer a framework including famous statistical formalisms for defining complex probability models such as Bayesian networks (BNs) (Pearl, 1988). These latter are directed acyclic graphs that allow to efficiently encode and manipulate probability distributions over high-dimensional spaces. BNs become quickly an important tool to address real world applications where uncertainty is almost an inescapable aspect. In 2011, the Turing prize was awarded to J. Pearl for the development of these models which reflect their important role for the industry as well as the scientific community. To perform probabilistic inference, the BN structure and parameters have to be already defined, if not, they have to be learned from data. Learning BN parameters consists on providing the conditional probability distributions CPDs of each variable given its parents in the graph. It can be done using maximum likelihood or Bayesian estimation methods. Learning the BN structure consists on providing a network structure which fits the best way to the observed data. This is a tremendous issue as the search space is usually very large and becomes quickly not scalable even with few number of variables. Several approaches have been proposed to tackle this issue and several manners have been proposed as well to evaluate the quality of the learning algorithms. The chapter is dedicated to introduce this framework and to make a survey on existing learning as well as evaluating approaches.

Contents

1.1 Introduction

Bayesian networks (Pearl, 1988) are founded on both graph and probability theories. They allow to deal with uncertainty through the use of probability theory. On the other hand, They allow to deal with the complexity of the addressed large spaces through the use of graph theory. Their construction implies the identification of both components: The graphical one which is a directed acyclic graph (DAG) representing probabilistic dependencies over a set of discrete random variables. The numerical one which is a set of conditional probabilistic distributions (CPDs). Several approaches have been proposed to address this task and several approaches have also been proposed to evaluate the quality of BNs learning methods.

The remainder of this chapter is as follows: Section 1.2 recalls some basic concepts from graph and probability theories. Section 1.3 defines Bayesian networks. Section 1.4 presents BNs learning approaches. Section 1.5 goes through evaluation techniques of BNs learning approaches. Section 1.6 presents a list of some existing Bayesian network-related softwares.

1.2 Basic concepts

Bayesian networks are founded on both graph and probability theories. Thus, it is useful to recall some useful concepts from these theories before addressing these models.

1.2.1 Useful concepts from graph theory

Definition 1.2.1. ***Graph.***

- *A graph is defined by:*
  - *A set ϑ of vertices or nodes and,*
  - *A set $E \subset \{(u,v) \| u,v \in \vartheta\}$ of edges or links.*
- *Two nodes that are connected by an edge are called adjacent.*
- *An edge may be undirected (unmarked link), directed (marked by a single arrowhead on the edge) or bidirected.*

Definition 1.2.2. ***Complete graph.***
A graph in which every pair of nodes is connected by an edge is a complete graph (e.g., Figure 1.1(a)).

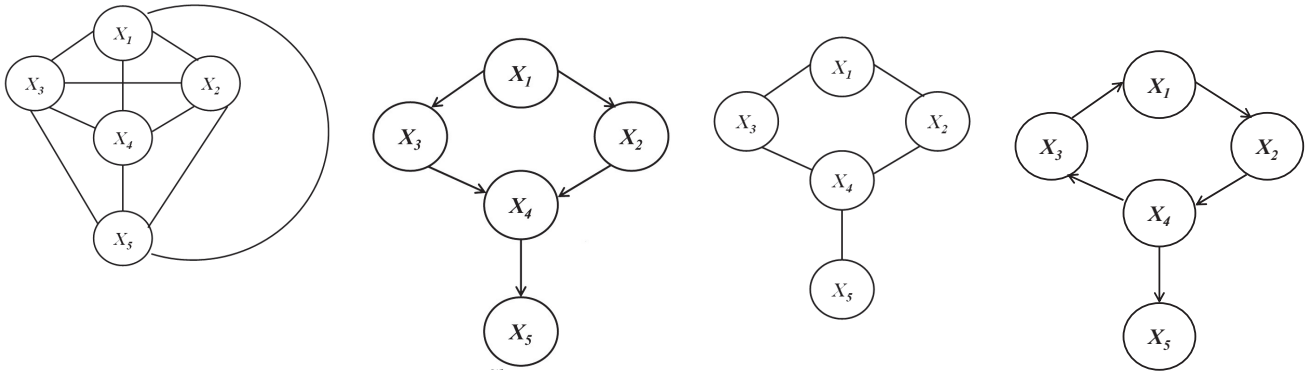
Definition 1.2.3. ***Undirected graph.***
A graph in which all edges are undirected is an undirected graph (e.g., Figure 1.1(c)).

Definition 1.2.4. ***Directed graph.***
A graph in which all edges are directed is a directed graph.

Definition 1.2.5. ***Skeleton of a directed graph.***
If we strip away all arrowheads from the edges of such a graph we obtain an undirected graph called its skeleton (e.g., Figure 1.1(c)).

Definition 1.2.6. ***Directed path.***
A directed path is a sequence of directed edges such that each edge starts with the vertex ending the preceding edge. (e.g., Figure 1.1(d): (X_1, X_2), (X_2, X_4), (X_4, X_5) is a path of length 3).

Definition 1.2.7. ***Directed cycle.***
A directed cycle is a directed path starting and ending at the same vertex (e.g., Figure 1.1(d): (X_1, X_2), (X_2, X_4), (X_4, X_3), (X_3, X_1) is a cycle). A cycle of length 1 (e.g., $X_1 \rightarrow X_1$) is a called a self-loop



(a) A complete graph (b) A directed acycle graph (DAG) (c) An undirected graph (d) A graph with cycle

Figure 1.1: Example of graphical models

Definition 1.2.8. ***Directed acyclic graph.***
A graph that contains only directed edges and no cycles is called directed acyclic graph (DAG)(e.g., Figure 1.1(b)).

- *The set of the ancestors $An(X_i)$ of a node X_i is the set of nodes that can reach X_i by a directed path of length one or more (e.g., Figure 1.1(b): $An(X_5) = \{X_1, X_2, X_3, X_4\}$).*
- *The set of the parents $Pa(X_i)$ of a node X_i, is the set of nodes that can reach X_i by a single arc pointing into X_i. A node with no parents is called a root (e.g., Figure 1.1(b): $Pa(X_5) = \{X_4\}$, X_1 is a root node).*
- *The set of the children $Ch(X_i)$ of a node X_i is the set of nodes reachable by a single arc from X_i. A node with no children is called a sink (e.g., Figure 1.1(b): $Ch(X_4) = \{X_5\}$, X_5 is a sink).*
- *The set of the descendants $De(X_i)$ of a node X_i is the set of nodes reachable by a directed path from X_i (e.g., Figure 1.1(b): $De(X_1) = \{X_2, X_3, X_4, X_5\}$).*
- *The set of the non-descendants $Nd(X_i)$ of a node X_i is defined as $Nd(X_i) = \vartheta \backslash (X_i \cup De(X_i) \cup Pa(X_i))$ (e.g., Figure 1.1(b): $Nd(X_2) = \{X_3\}$).*

1.2.2 Useful concepts from probability theory

Let us denote random variables by upper case letters (e.g., X, Y,...). The domain of any random variable X is denoted by $\mathcal{D}_X$. The assignments or states of X are denoted by corresponding lower-case letter x.

Definition 1.2.9. ***Joint probability distribution.***

The joint distribution of n random variables $X_1, \ldots, X_n$ is defined as follows:

$$\begin{aligned} P(X_1, \ldots, X_n): \quad & \mathcal{D}_{X_1} \times \ldots \times \mathcal{D}_{X_n} \rightarrow [0,1] \\ & (x_1, \ldots, x_n) \longmapsto P(x_1, \ldots, x_n) = P(\textstyle\bigcap_{i \in \{1,\ldots,n\}} \{X_i = x_i\}) \end{aligned} \tag{1.1}$$

To define the joint distribution of n random variables, we have to provide the Cartesian product of all the random variables domains. If each variable has k states, then there are k^n combinations. This product is exponential on the number of variables.

Definition 1.2.10. ***Conditional probability.***

Let X and Y be two random variables. $\forall x \in \mathcal{D}_X$ and $y \in \mathcal{D}_Y$. The conditional probability of $X = x$ under the condition $Y = y$ (evidence) and the conditional probability of $Y = y$ under the condition $X = x$ (evidence) are respectively expressed by:

$$P(x|y) = \frac{P(x,y)}{P(y)}, P(y) > 0 \tag{1.2}$$

and

$$P(y|x) = \frac{P(y,x)}{P(x)}, P(x) > 0 \tag{1.3}$$

*we can easily derive the **Bayes theorem** defined as follows:*

$$P(x|y) = \frac{P(y|x).P(x)}{P(y)}, P(y) > 0 \tag{1.4}$$

More generally:

$$P(x|y,z) = \frac{P(y|x,z).P(x|z)}{P(y|z)} \tag{1.5}$$

Definition 1.2.11. ***Marginal independence.***

Two random variables X and Y are called independent (we note: $X \perp Y$) if the outcome of X has no effect on the outcome of Y and vice versa.

Therefore:

$$X \perp Y \Leftrightarrow \left\{ \begin{array}{lcl} \forall x \in \mathcal{D}_X, & P(Y|X=x) & = & P(Y) \\ \forall y \in \mathcal{D}_Y, & P(X|Y=y) & = & P(X) \end{array} \right. \tag{1.6}$$

and the joint distribution of X and Y is:

$$P(X,Y) = P(X).P(Y) \tag{1.7}$$

Definition 1.2.12. ***Conditional independence.***

Let three random variables X, Y and Z. We say that X and Y are conditionally independent to Z (we note: $X \perp Y|Z$) if and only if:

$$X \perp Y|Z \Leftrightarrow \left\{ \begin{array}{lcl} P(X|Y,Z) & = & P(X|Z) \\ \textit{and } P(Y|X,Z) & = & P(Y|Z) \end{array} \right. \tag{1.8}$$

and the joint distribution of X, Y and Z is:

$$P(X,Y,Z) = P(X|Z).P(Y|Z).P(Z) \tag{1.9}$$

More generally, a joint distribution over n random variables may be expressed using conditional probabilities as follows:

$$P(X_1,\ldots,X_n) = \prod_{i=1}^{n}(P(X_i|X_1,\ldots,X_{i-1})) \tag{1.10}$$

This formula could be of interest when it is possible to simplify each $P(X_i|X_1,\ldots,X_{i-1})$ using conditional independence facts:

$$\forall i, V_i \subset \{X_1 \ldots X_{i-1}\}, \text{such that } X_i \perp (\{X_1 \ldots X_{i-1}\} \backslash V_i)|V_i, \quad P(X_1,\ldots,X_n) = \textstyle\prod_{i=1}^{n} P(X_i|V_i) \tag{1.11}$$

1.3 Bayesian network formalism

In this section, we give a formal definition of Bayesian networks (Pearl, 1988). Then, we recall the d-separation criterion. finally, we present the Markov equivalence class for directed acyclic graphs.

1.3.1 Bayesian network definition

Bayesian networks allow to get a compact encoding of a complex distribution over a high-dimensional space using a graph-based representation.

Formally, a Bayesian network (Pearl, 1988) (BN) $\mathcal{B} = (\mathcal{G}, \Theta)$ is defined by:

- A graphical (qualitative) component : a directed acyclic graph (DAG) $\mathcal{G} = (V, E)$, where V is the set of vertices (nodes) represents n discrete random variables $\mathcal{X} = \{X_1, \ldots, X_n\}$, and E is the set of directed edges (arcs) corresponds to conditional dependence relationships among these variables.
- A numerical (quantitative) component : presents a set of parameters $\Theta = \{\Theta_1, \ldots, \Theta_n\}$ where each $\Theta_i = P(X_i|Pa(X_i))$ denotes the conditional probability distribution of each node X_i given its parents $Pa(X_i)$.These conditional probability distributions are usually stored and organized in tables named conditional probability distributions (CPDs).

Example 1.3.1. *Figure 1.2 presents the graphical component of a Bayesian Network (Pearl, 2000). It illustrates a BN having five random variables and dependencies among them. X_1 is the season of the year and it can take one of four values: spring, summer, fall, or winter. All other variables are binary: X_2 describes whether the rain falls, X_3 describes whether the sprinkler is on, X_4 describes whether the pavement would get wet and X_5 describes whether the pavement would be slippery.*

1.3.2 Conditional Independence in Bayesian networks

A BN encodes the following conditional independence assumption, called *local Markov property*:

Property 1.3.1. ***(Local Markov Property)*** *For each variable X_i, we have that each variable X_i in $\mathcal{G}$ is independent of its non descendants $Nd(X_i)$ given its parents $Pa(X_i)$.*

$$X_i \perp Nd(X_i)|Pa(X_i)$$

Example 1.3.2. *In Figure 1.2, knowing X_4 makes X_5 independent of $\{X_1, X_2, X_3\}$.*

Pearl introduced a graphical criterion for determining whether two variables are independent conditionally to a set of variables, called the d-separation criterion (Pearl, 1988).

Before defining this criterion, let's present the three basic connection structures between variables:

- Serial connection (chain) ($X \rightarrow Z \rightarrow Y$) and diverging connection (fork) ($X \leftarrow Z \rightarrow Y$): If we know the value of Z , then X has no influence on Y and vice versa. We say that X and Y are dependent, but they are conditionally independent given Z, and we note: $X \perp Y|Z$.
- Converging connection (collider, also called V-structure) ($X \rightarrow Z \leftarrow Y$): If we know the value of Z, then evidence on X influences Y and vice versa. We say that X and Y are independent, but they are conditionally dependent given Z, and we note: $X \perp Y$ (but$X \not\perp Y|Z$).

Thus, the d-separation criterion is defined as follow (Pearl, 1988):

Definition 1.3.1. ***d-separation.***

*Let G = (V,E) be a DAG and let **X**, **Y**, and **Z** be disjoint sets of variables in V:*

1. *A path from some $X \in$ **X** to some $Y \in$ **Y** is d-connected given **Z** if and only if every collider W on the path, or a descendant of W, is a member of **Z** and there are no non-colliders in **Z**.*

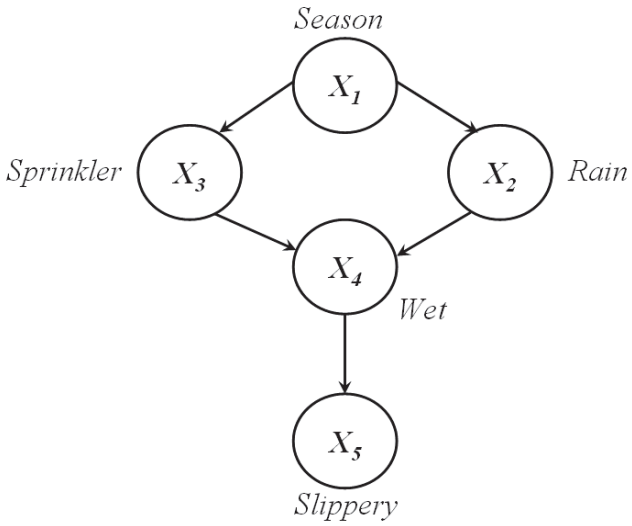


Figure 1.2: Example of a Bayesian network

2. ***X*** *and* ***Y*** *are said to be d-separated by* ***Z*** *if and only if there are no d-connecting paths between* ***X*** *and* ***Y*** *given* ***Z****.*

Example 1.3.3. *In Figure 1.2, $X = \{X_2\}$ and $Y = \{X_3\}$ are d-separated by $Z = \{X_1\}$ as both paths connecting X_2 and X_3 ($X_2 \leftarrow X_1 \rightarrow X_3$ and $X_2 \rightarrow X_4 \leftarrow X_3$) are blocked by Z (The first path presents a diverging connection having its middle node X_1 in Z. While the second path is a converging connection having its middle node X_4 out Z).*

Thanks to the local Markov property (cf. Property 1.3.1), Bayesian networks allow to represent the joint probability distribution (JPD) of $\mathcal{X} = \{X_1, \ldots, X_n\}$ as a decomposition of a global function into a product of local terms depending only on the considered node and its parents in the graph via the chain rule of BN:

$$P(\mathcal{X}) = \prod_{i=1}^{n} P(X_i | Pa(X_i)) \tag{1.12}$$

1.3.3 Markov equivalence class for Directed Acyclic Graphs

The Markov condition and the d-separation criterion connect the BN structure and conditional independence. When exactly the same set of d-separation relations hold in two directed graphs, we say that they are Markov equivalent.

Definition 1.3.2. ***Markov equivalence.***
Two Bayesian networks B_1 and B_2 are equivalent if they encode the same probability distribution.

Example 1.3.4. *Let's consider the Figure 1.3. In this example we can easy demonstrate that G1 and G2 are equivalent by the decomposition of their joint probability distributions (Naïm et al., 2004). We have:*

$$P(X_1, X_2, X_3)_{\mathcal{G}_1} = P(X_2|X_1) * P(X_1|X_3) * P(X_3)$$

and

$$P(X_1, X_2, X_3)_{\mathcal{G}_2} = P(X_2|X_1) * P(X_3|X_1) * P(X_1)$$

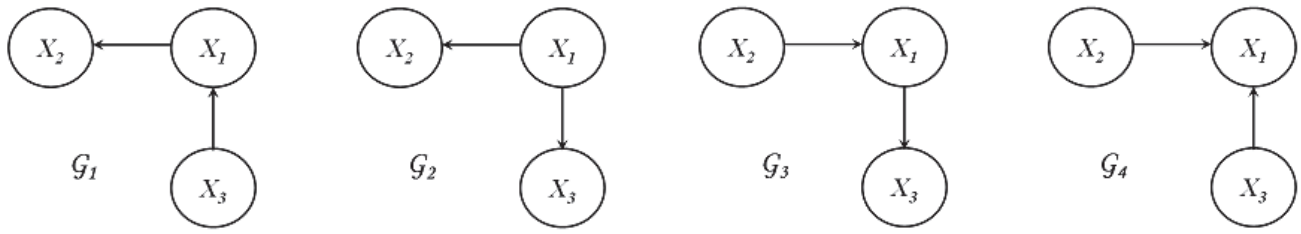


Figure 1.3: Markov equivalence

Thus:

$$\begin{aligned} P(X_1, X_2, X_3)_{\mathcal{G}_2} &= P(X_2|X_1) * P(X_3|X_1) * P(X_1) \\ &= P(X_2|X_1) * \frac{P(X_1|X_3)*P(X_3)}{P(X1)} * P(X_1) \\ &= P(X_2|X_1) * P(X_1|X_3) * P(X_3) \\ &= P(X_1, X_2, X_3)_{\mathcal{G}_1} \end{aligned}$$

Similarly, we demonstrate that G3 is equivalent to G1 and G2. Yet these three networks are not equivalent to the v-structure G4. As $P(X_1, X_2, X_3)_{\mathcal{G}_4} = P(X_1|X_2, X_3) * P(X_2) * P(X_3)$ *and* $P(X_1|X_2, X_3)$ *cannot be simplified.*

A graphical criterion for determining the equivalence of two DAGs was proposed by Verma and Pearl (Verma and Pearl, 1990) and expressed through the following theorem:

Theorem 1.3.1. *Two DAGs are equivalent if and only if they have the same skeletons and the same v-structures.*

A Markov equivalence class is defined as a set of equivalent Bayesian networks and represented via a Completed Partially Directed Acyclic Graph (CPDAG) (Andersson et al., 1997).

Definition 1.3.3. ***Completed Partially Directed Acyclic Graph.***

A CPDAG (essential graph) has the same skeleton as all the graphs in the equivalence class and all its reversible edges (edges that do not belong to a v-structure and their inversion does not generate a v-structure) are undirected.

Chickering (Chickering, 2002) proposes a method that allows the conversion of a DAG to the CPDAG representing its Markov equivalence class. Steps to construct the CPDAG are as presented by Algorithm 1. The algorithm starts by ordering all the BN edges using Algorithm 2. Then uses the set of ordered edges to simplify reversible edges.

Example 1.3.5. *For instance, the Completed Partially Directed Acycle Graph of* $\mathcal{G}_1$, $\mathcal{G}_2$ *and* $\mathcal{G}_3$ *of Example 1.3.4, is the undirected graph* $\mathcal{G} = X_2 - X_1 - X_3$.

1.3.4 Reasoning with Bayesian networks

Reasoning with Bayesian networks comes to perform probabilistic inference. This latter consists on inferring the whole probability distribution of some variables given that some other variables are set to certain values (evidence) (Pearl, 1988). A panoply of algorithms have been proposed to perform probabilistic inference (propagation) with BNs. Such a process looks for the impact of a certain information regarding some variables, known as an evidence, on the remaining ones. The first algorithms are based on message-passing architecture and were proposed by Pearl (Pearl, 1982), and Kim and Pearl (Kim and Pearl, 1983). An extension of these algorithms, that are limited to trees, is found in several works like the Lauritzen and Spiegelhalter's method of join-tree propagation (Lauritzen and Speigelhalter, 1988) and the method of

Algorithm 1 DAG_TO_CPDAG (Chickering, 2002)

Require: $\mathcal{G}$: A DAG
Ensure: $CPDAG_{\mathcal{G}}$: The Completed Partially Directed Acyclic Graph of the DAG

Order (E) % *The set of directed edges of* $\mathcal{G}$
$\forall e \in E, label(e) \leftarrow \emptyset$
$\mathcal{A} \leftarrow unlabeled(E)$
repeat
  $(X_i, X_j) \leftarrow min_A(e)$ % *Lowest unlabeled edge*
  $\forall K_k / label(X_k, X_i) \leftarrow irreversible$
  $End \leftarrow False$
  if $X_k \notin paX_j$ **then**
    $label(*, X_j) \leftarrow irreversible$
    $\mathcal{A} \leftarrow \mathcal{A} \setminus (*, X_j)$
    $End \leftarrow True$
  else
    $label(X_k, X_j) \leftarrow irreversible$
    $\mathcal{A} \leftarrow \mathcal{A} \setminus (X_k, X_j)$
  end if
  if $End = False$ **then**
    if $\exists e(X_k, X_j) / X_k \notin pa(X_i) \cup X_i$ **then**
      $\forall (X_k, X_j) \in \mathcal{A}$
      $label(X_k, X_j) \leftarrow irreversible$
      $\mathcal{A} \leftarrow \mathcal{A} \setminus (X_k, X_j)$
    else
      $\forall (X_k, X_j) \in \mathcal{A}$
      $label(X_k, X_j) \leftarrow reversible$
      $\mathcal{A} \leftarrow \mathcal{A} \setminus (X_k, X_j)$
    end if
  end if
until $\mathcal{A} = \emptyset$

Algorithm 2 Order (Chickering, 2002)

Require: E: list of edges, V: List of nodes
Ensure: $E_{ordered}$

$Topological\_Tree(X_i)$ % $X_i \forall \in V$
$k \leftarrow 0$
$\mathcal{A} \leftarrow unordered(E)$
repeat
  $X_j \leftarrow min_j(X_j / (X_i, X_j) \in \mathcal{A})$ % *Lowest destination node of an unordered edge*
  $X_i \leftarrow max_i(X_i / (X_i, X_j) \in \mathcal{A})$ % *Greatest source node of an unordered edge to* X_j
  $order(X_i, X_j) \leftarrow k$
  $k \leftarrow k + 1$
  $\mathcal{A} \leftarrow \mathcal{A} \setminus (X_i, X_j)$
until $\mathcal{A} = \emptyset$

cut-set conditioning (Pearl, 2000) that provide propagation in general networks. All these algorithms are exact, however, inference in general networks is known to be NP-hard (Cooper, 1990), which means that in some cases (e.g., a huge number of nodes, a densely connected DAGs) the computational complexity of these algorithms may exceed reasonable bounds thus, a set of approximate algorithms was proposed to be

used in such case. We refer readers to (Jensen and Nielsen, 2007; Darwiche, 2009; Larrañaga et al., 2013) for detailed information about these methods.

1.4 Bayesian networks structure learning

BN learning implies structure learning and parameter estimation from complete or incomplete training data. BN structure learning is known as an NP-Hard problem (Chickering et al., 1994). Once the structure is identified, parameters learning can be easily performed using either the statistical approach or the Bayesian one (Heckerman, 1998). In this section we focus on BN structure learning from complete data.

A wealth of literature has been produced that seeks to understand and provide methods of learning structure from data (Daly et al., 2011). BN structure learning methods are divided into three main families. The first family tackles this issue as a constraint satisfaction problem. Constraint-based algorithms look for independencies (dependencies) in the data, using statistical tests then, try to find the most suitable graphical structure with this information. The second family treats learning as an optimization problem. They evaluate how well the structure fits to the data using a score function. So, these Score-based algorithms search for the structure that maximizes this function. The third family presents hybrid approaches that mix both of the first two ones. In this section, we start by citing some learning assumptions. Then, we present the basics of these three families.

1.4.1 BNs learning assumptions

Besides the **local Markov property**, other assumptions are also required to perform structure learning, namely, the sufficiency and the faithfulness assumption.

1. **Sufficiency assumption:** The set of variables X is sufficient to represent all the conditional dependence relations that could be extracted from data (i.e., we assume the absence of latent variables).
2. **Faithfulness assumption:** We say that a joint probability distribution P and a DAG G are faithful to each other if and only if every independence present in P is entailed by G and the local Markov property.

1.4.2 Constraint-based approaches

Constraint-based approaches search for the best BN structure using conditional independencies obtained from statistical tests on the data. Several algorithms have been proposed in the literature, e.g., IC (Verma and Pearl, 1990), SGS (Spirtes et al., 1990), PC (Spirtes et al., 2000). All those algorithms share the same structure:

i. Construct an undirected dependency network from data.

ii. Detect V-structures.

iii. Try to orient more edges using the already oriented ones.

As all these approaches are based on reasoning about conditional independence facts, they learn the Markov equivalence class rather than the fully oriented BN structure.

The edge orientation step (iii) is based on a set of orientation rules. For instance, the PC algorithms uses the following three rules:

- V-structure detection: if $X - Y - Z$ and $Y \notin sepset(X, Z)$, then orient as $X \rightarrow Y \leftarrow Z$.
- Known non-V-structures: if $X \rightarrow Y - Z$ and $\langle X, Y, Z \rangle$ is not a V-structure, then orient as $X \rightarrow Y \rightarrow Z$.
- Cycle Avoidance: if $X - Y$ and $X \rightarrow V_1 \ldots \rightarrow V_k \rightarrow Y$, then orient as $X \rightarrow Y$.

Given sufficiency and faithfulness assumptions (cf. Section 1.4.1), it has been proved that the result of the PC algorithm converges to the CPDAG of the true model if statistical tests are substituted with an oracle function able to provide dependencies derived from applying d-separation on the true model (Spirtes et al., 2000).

Constraint-based methods are efficient and work well with sparse graphs, yet they are sensitive to failures in independence tests. Thus, score-based methods are commonly known to be a better tool for learning structure from data (Friedman et al., 1999a).

1.4.3 Score-based approaches

Score-based approaches, also known as 'score-and-search' techniques have been widely studied to learn BNs structure. Several algorithms have been proposed such as Maximum Weight Spanning Tree(MWST) (Chow and Liu, 1968), K2 (Cooper and Herskovits, 1992), Greedy Search (GS), Greedy Equivalence Search (GES) (Heckerman, 1998; Chickering, 2002; Chickering and Maxwell, 2002), etc. All proposed methods present:

i. a search space allowing to determine which are the candidate legal Bayesian networks,

ii. a scoring function to assign a score to each candidate structure in the search space and,

iii. an effective search procedure to move from structure to another in the search space.

1.4.3.1 The search space

Each score-based algorithm defines a hypothesis space of potential models, namely, the set of possible network structures it is willing to consider. Some algorithms are limited to small spaces such as trees (Chow and Liu, 1968), polytrees and hypertrees (Srebro, 2001). Other approaches search either through the space of equivalence classes, known as the $\mathcal{E}$-space (Chickering, 2002), or through the space of DAGs, known as the $\mathcal{B}$-space (Cooper and Herskovits, 1992).

1.4.3.2 The scoring function

The idea over these approaches is to find the network that fits best to data while being as simple as possible. Thus, score-based algorithms assign a score to each possible network based on these two criteria, to search after through the space of graphs for the network that maximizes this score. Several scoring functions have been developed to assess the goodness-of-fit of a particular model (e.g., Bayesian information criterion (BIC) (Schwarz, 1978), Bayesian Dirichlet (BD) (Cooper and Herskovits, 1992), Akaike information criterion (AIC) (Akaike, 1970), minimum description length (MDL) (Bouckaert, 1993)). All scoring functions share two interesting properties:

- **Score-equivalent scoring function:** A scoring function is score-equivalent if it assigns the same score to equivalent structures (cf. Definition 1.3.2).
- **Decomposable scoring function:** A scoring function is decomposable if it can be written as a sum of measures, each of which is a function only of one node and its parents.

For instance, the BD score presented in (Cooper and Herskovits, 1992) as part of the $K2$ algorithm consists of two main parts: The prior probability of the structure $P(\mathcal{G})$ and the likelihood of the structure given the data $P(\mathcal{G}|D)$.

Formally, the BD score is expressed by:

$$BD_{score} = P(\mathcal{G}, D) = P(\mathcal{G}) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(N_{ij} + \alpha_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N_{ijk} + \alpha_{ijk})}{\Gamma(\alpha_{ijk})} \tag{1.13}$$

where n is the number of variables, q_i is the possible parent configurations of variable i, r_i is the number of values variable i, N_{ijk} is the number of times where variable i took on value k with parent configuration j $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$ and Γ is the gamma function. However, the BD score is not score-equivalent.

Buntine (Buntine, 1991) proposed the Bayesian Dirichlet Equivalent score known as BDeu. This latter uses the same formula as the BD score, with Dirichlet coefficients:

$$\alpha_{ijk} = \frac{N'}{r_i q_j} \tag{1.14}$$

where N' is a user defined value to express the number of equivalent examples.

Heckerman et al. (Heckerman et al., 1995) proposed another variant of the BDeu score known as the BDe score, with Dirichlet coefficients:

$$\alpha_{ijk} = N' \times P(X_i = x_k, Pa(X_i = x_j)|\mathcal{G}_c) \tag{1.15}$$

where $\mathcal{G}_c$ is the complete graph.

In the case where X_i has an uniform conditional probability distribution, α_{ijk} collapse to the Dirichlet coefficients of Equation 1.14, that corresponds to an uniform, non-informative prior. An interesting property of the BDeu and BDe scores is that they assign the same score for equivalent structures.

1.4.3.3 The search procedure

Having a set of possible network structures, a scoring function and a training dataset, we are dealing with an optimization problem, where the desired output is a network structure, from the set of possible structures, that maximizes the score. In the general case, finding an optimally scoring $\mathcal{G}*$ is NP-hard. Most score-based methods use classical heuristic techniques that attempt to find the highest scoring structure, but are not guaranteed to do so.

The simplest search procedure is the greedy one. Greedy search (GS) performs as follows: for each BN structure $\mathcal{G}$, GS moves on to the neighbor graph that has a better score, until it reaches a structure that has the highest score in the list of neighbors. A natural choice of neighbors of a BN structure is a set of structures that are identical to it except for small local modifications. The most commonly used operators which define the local modifications are add an edge, delete an edge and reverse an edge, while considering operations that result in legal networks (i.e., DAGs).

One of the simplest and often used search procedures is the greedy hill-climbing procedure (Heckerman, 1998; Chickering, 2002), based on 6 steps as follows: 1) Start from an initial network structure $\mathcal{G}$ that may be either empty or not (e.g., obtained from some prior knowledge). 2) Compute its score. 3) Consider all of the neighbors of $\mathcal{G}$ in the space (by applying a single operator to $\mathcal{G}$ (cf. Algorithm 4)). 4) Compute the score for each of them. 5) Apply the change that leads to the best improvement in the score. 6) Continue this process until no modification improves the score. Algorithm 3 summarizes all these steps.

Score-based methods work better with less data than constraint-based methods and with probability distributions that admit dense graphs. However, as the search space is extremely large, score-based approaches spend a lot of time at examining candidates, and this problem becomes more awkward when we deal with massive data (i.e., large in number of instances and attributes). Both constraint-based and score-based approaches present some advantages and suffer from some drawbacks, thus researchers have tried to draw hybrid methods on the basics of the use of the good points of both approaches.

1.4.4 Hybrid approaches

Hybrid algorithms combine the main features of both techniques by using the local conditional independence tests and the global scoring functions. The idea was at the beginning provided by (Singh and Valtorta, 1993; Singh and Valtorta, 1995). First, they construct a total ordering of the variables using conditional independence tests. Then, they use this ordering as input to the $K2$ algorithm to learn the structure. Several works have followed this technique (e.g., (Provan and Singh, 1995; Acid and de Campos, 1996; Acid and de Campos, 2000; Acid and de Campos, 2001)).

Algorithm 3 Greedy hill-climbing (Heckerman, 1998)

Require: $\mathcal{D}$: A database, $\mathcal{G}$: Initial BN structure, $Score$: A scoring function
Ensure: $\mathcal{G}$: The local optimal BN structure

$Max_{score} \leftarrow Score(\mathcal{G})$
$No\_Changes \leftarrow false$
repeat
  $List\_neighbors \leftarrow Generate\_neighborhood(\mathcal{G})$
  $\mathcal{G}_{new} \leftarrow Argmax_{\mathcal{G}' \in neighborhood_{\mathcal{G}}}(Score(\mathcal{G}'))$
  if $Score(\mathcal{G}_{new}) \geq Max_{score}$ **then**
    $Max_{score} \leftarrow score(\mathcal{G}_{new})$
    $\mathcal{G} \leftarrow \mathcal{G}_{new}$
  else
    $No\_Changes \leftarrow true$
  end if
until No_Changes

Algorithm 4 $Generate\_neighborhood$

Require: $\mathcal{G}$: A BN structure
Ensure: $\mathcal{N}$: A list of neighbors DAGs

$\mathcal{N} \leftarrow \emptyset$
for all $e \in \mathcal{G}$ **do**
  $\mathcal{N} \leftarrow \mathcal{N} \cup (\mathcal{G} \setminus \{e\})$ % *delete_edge(e).*
  if acyclic $(\mathcal{G} \setminus \{e\} \cup invert(e))$ **then**
    $\mathcal{N} \leftarrow (\mathcal{N} \cup \mathcal{G} \setminus \{e\} \cup invert(e))$ % *invert_edge(e).*
  end if
end for
for all all $e \notin \mathcal{G}$ **do**
  if acyclic $(\mathcal{G} \cup \{e\})$ **then**
    $\mathcal{N} \leftarrow \mathcal{N} \cup (\mathcal{G} \cup \{e\})$ % *add_edge(e).*
  end if
end for

(Dash and Druzdzel, 1999) combine the PC algorithm with GS. (Acid and de Campos, 2003) do the opposite: they perform an initial GS with random restart and then use conditional independence tests to add and delete arcs from the obtained DAG.

Friedman et al. (Friedman et al., 1999b) proposed the *Sparse Candidate* (SC) algorithm. It uses conditional independence to find good candidate parents and hence limit the size of the search in later stages. This algorithm is very useful in increasing the speed of search procedures, without unduly damaging the score. Tsamardinos et al. proposed the max-min-hill-climbing (MMHC) algorithm considered as an instantiation of the SC algorithm. $MMHC$ outperforms state-of-the-art learning algorithms and is applied to large datasets (Tsamardinos et al., 2006).

MMHC is an hybrid approach that combines the local search technique of the constraint-based methods and the overall structure search technique of the score-based methods. This algorithm consists of two phases

- The first phase aims to find, for each node in the graph, the set of candidate nodes that can be connected to it. At this stage there is no distinction between children and parents nodes and links orientation is not of interest.
- The second phase allows the construction of the graph $\mathcal{G}$ using the greedy search heuristic constrained to the set of candidate children and parents of each node in the graph as defined in the first phase.

The first step is described by Algorithm 5 and Algorithm 7, while the overall process is summarized in

Algorithm 5 $\overline{MMPC}$ (Tsamardinos et al., 2006)

Require: $\mathcal{D}$: A database, T: A target variable, Pot_{list}: The list of potential neighbors of T
Ensure: CPC: The set of candidate parents and children of T *% Phase I: Forward*

$CPC \leftarrow \emptyset$
repeat
  $\langle F.assocF \rangle \leftarrow MaxMinHeuristic(T, CPC, Pot_{list})$
  if $assocF \neq 0$ **then**
    $CPC \leftarrow CPC \cup F$
    $Pot_{list} \leftarrow Pot_{list} \backslash F$
  end if
until CPC has not changed or $assocF = 0$ or $Pot_{list} = \emptyset$
% Phase II: Backward
for all $X \in CPC$ **do**
  if $\exists S \subseteq CPC, s.t. Ind(X; T|S)$ **then**
    $CPC \leftarrow CPC \backslash \{X\}$
  end if
end for

Algorithm 6 MaxMinHeuristic (Tsamardinos et al., 2006)

Require: T: A target variable, CPC: A subset of variables, Pot_{list}: The list of potential neighbors of T
Ensure: F: The variable that maximizes the minimum association with T relative to CPC, $assocF$: F association measurement

$assocF \leftarrow max_{X \in Pot_{list}} MinAssoc(X; T|CPC)$
$F \leftarrow argmax_{X \in Pot_{list}} MinAssoc(X; T|CPC)$

Algorithm 7 $MMPC$ (Tsamardinos et al., 2006)

Require: $\mathcal{D}$: A database, T: A target variable
Ensure: CPC: The set of candidate parents and children of T

$Pot_{list} \leftarrow \mathcal{V} \backslash T$
$CPC \leftarrow \overline{MMPC}(T, \mathcal{D}, Pot_{list})$
for all $X \in CPC$ **do**
  if $T \notin \overline{MMPC}(X, \mathcal{D}, \mathcal{V} \backslash X)$ **then**
    $CPC \leftarrow CPC \backslash \{X\}$
  end if
end for

Algorithm 8 $MMHC$ (Tsamardinos et al., 2006)

Require: $\mathcal{D}$: A database, $Score$: A scoring function
Ensure: BN: A BN on the set of variables $\mathcal{V}$ in $\mathcal{D}$

$All\_CPCs \leftarrow \emptyset$ *% Local search*
for all $X \in \mathcal{V}$ **do**
  $CPC_X \leftarrow MMPC(X, \mathcal{D})$
  $All\_CPCs \leftarrow All\_CPCs \cup CPC_X$
end for
% Global search
$BN \leftarrow Greedyhill - climbing(\mathcal{D}, BN, Score, All\_CPCs)$

Algorithm 8. The MMPC algorithm (cf. Algorithm 7) discovers the set of candidate parents and children

(CPC) for a target variable T. It consists of the $\overline{MMPC}$ (Algorithm 5) and an additional symmetrical correction. $MMPC$ removes from each set $CPC(T)$ each node X for which $T \notin CPC(X)$.

The $MMPC$ algorithm states that two variables X and Y are considered as neighbors if $Y \in CPC(X)$ AND $X \in CPC(Y)$. (de Morais and Aussem, 2010) refer to this assumption as the *AND condition*. This condition has been used by several algorithms such as MMPC, MMMB (Tsamardinos et al., 2006), PCMB (Peña et al., 2007). yet, (de Morais and Aussem, 2010) considers this condition as severe and may yield too many false negatives in the result, especially when the sample size is limited. (de Morais and Aussem, 2010) provide a conservative approach that gives more chance for true positive to enter the $CPC(T)$ set under practical sample size limitations. They proposed the Markov Boundary search using the OR condition (MBOR) algorithm, a Markov boundary learning algorithm under the *OR condition* which states that two variables X and Y are considered as neighbors if $Y \in CPC(X)$ OR $X \in CPC(Y)$. The *OR condition* gives more opportunity for true positive nodes to enter the Markov boundary. The MBOR algorithm is scalable and correct under the faithfulness condition (cf. Section 1.4.1). In addition, experimental results have shown a notable benefit of the *OR condition* in case of densely connected DAGs or weak associations (de Morais and Aussem, 2010).

$\overline{MMPC}$ (Algorithm 5) consists of a forward phase where for each variable T of the graph, a set of variables are added to $CPC(T)$, and a backward phase whose role is to remove false dependencies detected in the forward phase. During the forward phase. $MaxMinHeuristic$ selects the variables that maximize the $MinAssoc$ with target variable T conditioned to the subset of the currently estimated CPC. $MinAssoc$ estimates the strength of association between two variables given CPC. Dependency is measured using an association measurement function such as mutual information, $\mathcal{G}^2$, χ^2.

Tsamardinos et al. assume independence without performing independence tests if there are less than five training instances on average per parameter (count) to be estimated. Contrary to Spirtes et al. (Spirtes et al., 2000) who assume dependence if there are too few records. (de Jongh and Druzdzel, 2014) refers to these rules as the *remove* and *keep* rules respectively. They made an experimental comparison between these two assumptions and they experimentally proved that the remove rule outperforms the keep rule as it allows to minimize the structural distance between the true BN structure and the learned one.

Algorithm 8 performs a Greedy hill-climbing search (Algorithm 3). The main difference is that the $Generate\_neighborhood$ procedure (Algorithm 4) will generate the neighbors list $\mathcal{N}$ from only the sets of candidate parents and children detected by Algorithm 7.

Tsamardinos et al. (Tsamardinos et al., 2006) provided a wide comparative study among several algorithms from three algorithm families, using several benchmarks and metrics (execution time, SHD measure, etc.) Following this study, they showed that hybrid approaches provide better results than other ones.

The next section provides more details on benchmarks and metrics used in the literature to evaluate structure learning algorithms.

1.5 Evaluating Bayesian networks structure learning algorithms

The evaluation process of a BN structure learning algorithm often requires the existence of a theoretical (also called gold) network from which one can sample a training data. Then learning is performed using this sampled data and the learned network is compared to the theoretical one using some evaluation metrics. The overall process is shown by Figure 1.4.

Usually, famous BNs are used in the literature as gold nets, yet, some researchers use a random generation process of synthetic BNs in order to provide a large number of possible models and to carry out experimentation while varying models from simple to complex ones, depending on the addressed learning area. This section presents some famous Bayesian networks used in the literature, describes some BN random generation processes, introduces some sampling approaches and goes through existing evaluation measures.

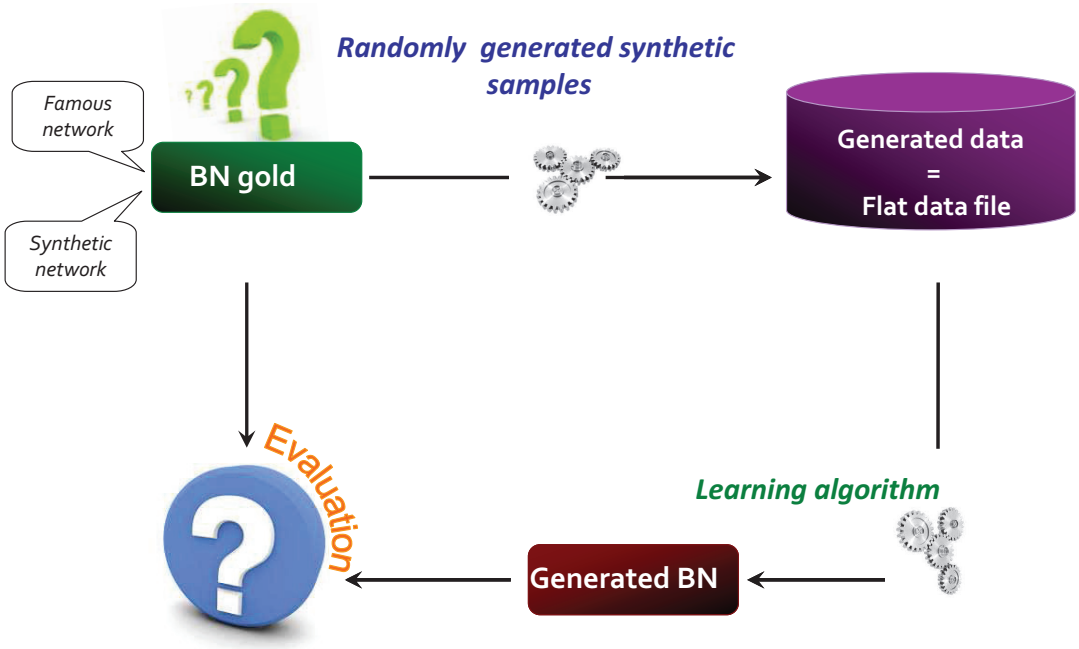


Figure 1.4: Evaluation process of a BN structure learning algorithm

1.5.1 Gold Bayesian networks

1.5.1.1 Famous gold Bayesian networks

BNs have proved their efficiency in several domains such as medicine, industry, marketing, security, etc (Naïm et al., 2004). Some Bayesian networks used in these domains have been reused later by other researchers or shipped as examples by BN software tools. This continuous use made them famous in the literature, and they have become often used to evaluate new approaches related to probabilistic inference in BNs or the generation of these models and to compare reported results to previous work results.

Famous gold BNs can be classified according to the application domain from which they were derived or their size as they varies from small (i.e., less than 20 nodes) to massive ones (more than 1000 nodes). Here are some examples of famous BNs:

The Asia network, also called *LUNG CANCER network*. It is a small network with 8 nodes representing 8 binary random variables and 8 arcs. It has been initially defined by (Lauritzen and Speigelhalter, 1988) as a fictitious medical knowledge example and is considered as a typical diagnostic network. It provides a diagnosis of a patient, having just come back from Asia and showing dyspnoea.

The Insurance network, with 27 nodes and 52 arcs, the insurance network is considered as a medium net. It has been introduced by Binder et al. (Binder et al., 1997) and used to classify car insurance applications.

The MUNIN network, the Muscle and Nerve Inference Network has been introduced by Andreassen et al. (Andreassen et al., 1989). It allows to diagnose peripheral muscle and nerve diseases. This network is considered as a massive network since it contains 1041 nodes and 1397 arcs.

1.5.1.2 Random generation of gold Bayesian networks

(Statnikov et al., 2003) proposed an algorithmic approach to generate arbitrarily large BN by tiling smaller real-world known networks. The complexity of the final model is controlled by the number of tiling and a connectivity parameter which determines the maximum number of connections between one node and the next tile. Some works have been devoted to the generation of synthetic networks but without guarantees that every allowed graph is produced with the same uniform probability (Ide and Cozman, 2002). In (Ide et al., 2004) the authors have proposed an approach allowing to generate uniformly distributed Bayesian

Algorithm 9 PMMixed (Ide et al., 2004)

Require: n: Number of nodes, I: number of iterations,
W: maximum induced width, O: possibly constraints on node degree and number of nodes
Ensure: A connected DAG with n nodes

Create a network with n nodes, where all nodes have just one parent, except the first node that does not have any parent.
for $i \leftarrow 1$ to I **do**
  if current graph is a polytree **then**
    With probability p, call Procedure AorR(current graph)
    with probability $1-p$, call Procedure AR(current graph)
    if the resulting graph satisfies W and O **then**
      accept the graph
    else
      keep previous graph
    end if
  else
    Call Procedure AorR (current graph)
    if the resulting graph is a polytree and satisfies W and O **then**
      accept with probability p
    else
      if it satisfies W and O **then**
        accept the graph
      else
        keep previous graph
      end if
    end if
  end if
end for

Algorithm 10 Add and Remove (AR) (Ide et al., 2004)

Require: $\mathcal{G}_{initial}$: An initial DAG
Ensure: $\mathcal{G}$ A DAG

Generate uniformly a pair of distinct nodes i, j
if the arc (i, j) exists in $\mathcal{G}_{initial}$ **then**
  $\mathcal{G} \leftarrow \mathcal{G}_{initial}$
else
  Invert the arc with probability $1/2$ to (j, i)
  Find the predecessor node k in the path between i and j
  $\mathcal{G} \leftarrow \mathcal{G}_{initial} \setminus \{(k, j)\}$ *%Remove the arc between k and j.*
  $\mathcal{G} \leftarrow \mathcal{G} \cup \{(i, j)\}$ OR $\mathcal{G} \leftarrow \mathcal{G} \cup \{(j, i)\}$*%Add an arc (i, j) or arc (j, i) depending on the result of the inversion*
end if

networks using Markov chains, known as the PMMixed algorithm. Using this algorithm, constraints on generated nets can be added with relative ease such as adding constrains on nodes degree, maximum number of dependencies in the graph, etc. This is done by constructing an ergodic Markov chain with uniform limiting distribution, such that every state of the chain is a DAG satisfying the constraints. After a given number of iterations, the algorithm leads to a satisfactory DAG. The overall process is outlined by Algorithm 9 that calls the two procedures AR (Algorithm 10) and AorR (Algorithm 11).

Algorithm 11 Add or Remove (AorR) (Ide et al., 2004)

Require: $\mathcal{G}_{initial}$: An initial DAG
Ensure: $\mathcal{G}$: A DAG
1: Generate uniformly a pair of distinct nodes i, j
2: **if** the arc (i, j) exists in $\mathcal{G}_{initial}$ **then**
3: $\mathcal{G} \leftarrow \mathcal{G}_{initial} \setminus \{(i, j)\}$ *%Delete the arc.*
4: **if** $\mathcal{G}$ does not remain connected **then**
5: $\mathcal{G} \leftarrow G \cup \{(i, j)\}$ *%Add the arc.*
6: **end if**
7: **else**
8: $\mathcal{G} \leftarrow \mathcal{G}_{initial} \cup \{(i, j)\}$ *%Add the arc.*
9: **if** $\mathcal{G}$ does not remain acyclic **then**
10: $\mathcal{G} \leftarrow \mathcal{G} \setminus \{(i, j)\}$ *%Delete the arc.*
11: **end if**
12: **end if**

Algorithm 12 Forward Sampling (Henrion, 1986)

Require: G: A Bayesian network over $\mathcal{X} = \{X_1, \ldots, X_n\}$ nodes, $\mathcal{N}$: The number of desired samples
Ensure: A sample of size $\mathcal{N}$
1: $\delta \leftarrow$topological ordering of $\mathcal{X}$
2: **for** $i \leftarrow 1$ to $\mathcal{N}$ **do**
3: **for** $j \leftarrow 1$ to $|\delta|$ **do**
4: $u_j \leftarrow x(Pa(X_i))$ $_{//(Pa(X_j)) \in \{x_1, \ldots x_{j-1}\}}$
5: sample x_j from $P(X_j|u_j)$
6: **end for**
7: **end for**

1.5.2 Sampling Bayesian networks

Having either a famous Bayesian network or a generated BN resulting from a random generation process (i.e., the BN structure and parameters), a sampling method is applied to generate observational data and evaluate the ability of the learning algorithm to find the initial net from which data has been sampled (e.g., Metropolis-Hastings MCMC (Metropolis et al., 1953), Forward Sampling (Henrion, 1986), Likelihood Weighting (Fung and Chang, 1990), Gibbs Sampling (MCMC) (Gelfand and Smith, 1990; Smith and Roberts, 1993)). The Forward Sampling is a well known sampling technique: nodes are sampled with respect to the topological order of the ground graph, so that by the time we sample a node we have values for all of its parents. After, we sample from the defined distribution and by the chosen values for the node's parents (cf. Algorithm 12).

1.5.3 Evaluation metrics

The evaluation of any learning algorithm is made according to two types of evaluation measures: the first one is related to the execution speed while the second examines the quality of reconstruction.

1.5.3.1 Evaluating the execution speed

The execution speed is measured with respect to the execution time and the number of statistical calls performed by an algorithm. The first metric is closely related to the specific implementation. So, to have faithful results using this metric it would be better to use this metric with the same implementation environment for the set of compared algorithms. The second metric computes the total number of statistical calls

used by an algorithm. Depending on the considered algorithm family, these measures might be either tests of independence, or measures of association or local scores.

1.5.3.2 Evaluating the quality of reconstruction

Two complementary ways can be used to evaluate the capacity for reconstruction, namely reconstructing the graphical structure and reconstructing the associated joint probability distribution (de Campos, 2006).

Evaluating the graph reconstruction. This evaluation returns to measure the structural differences between the original and the learned networks by computing the number of added, deleted and inverted edges in the learned network with respect to the original one. Methods to evaluate the graph reconstruction quality may be divided into three families: a) methods based on sensitivity and specificity, b) methods based on score functions and c) methods based on distance measures.

a) Sensitivity and specificity-based method. Sensitivity and specificity are statistical measures of the performance of a binary classification test, also known as classification function in statistics (FAST, 2010).

Sensitivity measures the proportion of actual positives which are correctly identified, while specificity measures the proportion of negatives which are correctly identified.

Given the graph of the theoretical network $\mathcal{G}_0 = (V, E_0)$ and the learned graph $\mathcal{G} = (V, E)$, The sensitivity-specificity-based method begins by calculating the following values:

- TP (true positive) = number of edges present in both $\mathcal{G}_0$ and $\mathcal{G}$.
- TN (true negative) = number of edges absent in both $\mathcal{G}_0$ and $\mathcal{G}$.
- FP (false positive) = number of edges present in $\mathcal{G}$, but not in $\mathcal{G}_0$.
- FN (false negative) = number of edges absent in $\mathcal{G}$, but not in $\mathcal{G}_0$.

Then, the sensitivity and specificity can be calculated as follows:

$$Sensitivity = \frac{TP}{TP + FN} \tag{1.16}$$

$$Specificity = \frac{TN}{TN + FP} \tag{1.17}$$

These measures are easy to calculate. They are often used in the literature. However, the differences in orientation between two graphs in the same equivalence class are counted as errors, even if they are not distinguishable from data.

b) Score-based method. To evaluate a structure learning algorithm, many studies use a comparison between the score function of the learned graph and the original graph. The learning algorithm is good if $S(\mathcal{G}, D) \cong S(\mathcal{G}_0, D)$ where S is a score function. One interesting property of score functions is that they assign the same score to networks that share the same essential graph. Nevertheless, the use of scores to evaluate the reconstruction quality raises two criticisms: first, the score corresponds to the a posteriori probability of a network only under certain conditions (e.g., a Dirichlet distribution of the hyperparameters for the $BDeu$ score) and it is unknown to what degree these assumptions hold in distributions encountered in practice. Second, in practice, score functions are very sensitive with respect to the equivalent sample size used.

c) Distance-based method. Tsamardinos et al. (Tsamardinos et al., 2006) adapted the Structural Hamming Distance, usually used for DAGs, (SHD) for $PDAGs$.

Definition 1.5.1. ***Structural Hamming Distance.***
The Structural Hamming Distance between two $PDAGs$ is defined as the number of the following operators required to make the $PDAGs$ match: add or delete an undirected edge, and add, remove, or reverse the orientation of an edge.

Algorithm 13 SHD (Tsamardinos et al., 2006)

Require: H: Learned PDAG, G: True PDAG
Ensure: shd value
1: $shd \leftarrow 0$
2: **for** every edge E different in H than G **do**
3: **if** E is missing in H **then**
4: $shd+ = 1$
5: **end if**
6: **if** E is extra in H **then**
7: $shd+ = 1$
8: **end if**
9: **if** E is incorrectly oriented in H **then**
10: $shd+ = 1$
This includes reversed edges and edges that are undirected in one graph and directed in the other.
11: **end if**
12: **end for**

The SHD of a given learned PDAG is computed as follows: An extra unoriented edge is penalized by an increase of the distance by 1 and not orienting an edge that should be oriented is penalized by an increase of the distance by 1 too. SHD is applied to $PDAG$ to avoid penalizing structural differences that cannot be statistically distinguished. Thus, for learning algorithms that return $DAGs$, the $PDAG$ has to be generated before using the SHD metric (cf. Algorithm 1). The overall steps of the SHD metric computation are described by Algorithm 13.

Evaluating the joint probability distribution reconstruction. To reconstruct the joint probability distribution, we can estimate the Kullback-Leibler divergence or KL-divergence (Kullback and Leibler, 1951) between the distributions associated with the original and the reconstructed networks. This measure does not directly penalize for extraneous edges and parameters (Tsamardinos et al., 2006).

1.6 Bayesian network-related softwares

Several software tools have been developed for Bayesian networks. Some of them are commercial while some others are open source. Here we present a non exhaustive list of existing tools and we describe briefly the functionalities allowed by each of them.

1.6.1 Commercial software tools

HUGIN. [1] a well known model-based decision support software for reasoning under uncertainty. It uses several models to support decision making including Bayesian network. Hugin Developer comprises the Hugin GUI and the Hugin Decision Engine. The user interface contains a graphical editor allowing to easily construct and maintain BNs. The Hugin Decision Engine contains all functionalities related to handling and using knowledge bases in a programming environment. Five major programming languages are supported, namely, C, C++, Java, .NET and an ActiveX-server.

Functionalities related to the definition, inference and learning of Bayesian networks are all developed. Several inference and learning algorithms are supported and BNs with continuous variables as well as learning from missing data are treated.

1. http://www.hugin.com

Netica. [2] a commercial software tool with a reduced price for the scientific community. Its intuitive user interface allows to draw the networks. Parameters are then easily introduced or learned from data, with support of learning from missing data. Several inference algorithms are supported too. Structure learning is supported from version 5.0 and later, using the TAN Structure learning functionality.

BayesiaLab. [3] a french software tool distributed by Bayesia. BayesiaLab handles exact and approximate inference. BNs learning is also so developed in this software. It provides parameter learning using maximum likelihood estimation and a variety of structure learning algorithms. It supports missing data, prior knowledge, supervised learning, etc.

Probayes. [4] a french software tool developed using the C++ language. It provides solutions in areas of decision support using artificial intelligence and machine learning techniques. For non-commercial purposes, Probayes provides the ProBT library [5]. ProBT is a set of high-performance algorithms. It allows to create and manipulate BNs objects and provides a set of learning and inference algorithms.

1.6.2 Open source software tools

Bayes Net Toolbox for Matlab. [6] Initially provided by Kevin Murphy in 1977, the BNT toolbox supports many different inference algorithms either exact or approximate. It also supports several methods for parameter learning from either complete or incomplete data sets. Structure learning can also be performed as a number of known algorithms (e.g., MCMC, IC, PC) are already developed under the BNT toolbox. Additionally, the source code is extensively documented which constitutes a strength for this tool and explains its wide use, especially by the scientific community.

Bayesian network structure learning, parameter learning and inference for R. [7] bnlearn is a R package for BN parameter and structure learning. It is also useful to perform probabilistic inference. The package provides several score-based, constraint-based and hybrid structure learning algorithms and supports discrete and continuous data sets. It supports statistical and Bayesian parameter estimation methods. In addition, it provides some other utility functions, such as random data generation.

OpenMarkov. [8] Developed by the Research Center for Intelligent Decision-Support Systems of the National Distance Education University (UNED) in Madrid. OpenMarkov is an open source tool, developed in Java. The graphical user interface (GUI) has two working modes: edition and inference. OpenMarkov offers two structure learning algorithms: the PC algorithm, as a constraint-based method and the hill climbing algorithm, as a score-based algorithm. The default implementation of parameter learning is a Bayesian approach using a Laplace-like correction.

Mens X Machina Probabilistic Graphical Model Toolbox (MxM PGM). [9] Is a MATLAB toolbox to manipulate Bayesian networks and other probabilistic graphical models. Currently, MXM provides source code only for the local Bayesian network learning and Bayesian network skeleton identification using the MMPC algorithm (cf. Algorithm 7).

2. www.norsys.com/
3. http://www.bayesia.com
4. http://www.probayes.com
5. http://www.probayes.com/fr/Bayesian-Programming-Book/downloads/
6. bnt.googlecode.com
7. http://www.bnlearn.com
8. http://www.openmarkov.org
9. http://www.ics.forth.gr/bil/index_main.php?l=e&c=480

1.7 Conclusion

In this chapter, we introduced basic definitions and concepts related to Bayesian networks. Then, we described the current existing approaches for BN structure learning from complete data. We have seen that hybrid methods perform better than other techniques based only on conditional independence tests or on score functions on the level of scalability and complexity of algorithms. We also addressed the BN structure learning evaluation task which consists of random model generation, sampling and evaluation metrics.

The use of BNs as a knowledge representation tool increases more and more in the machine learning community, and several extensions have been proposed in order to broaden their range of application (e.g., Causal Bayesian Networks (Pearl, 2000) (CBN) Dynamic Bayesian Networks (Murphy, 2002) (DBN), Hierarchical Bayesian Networks (Gyftodimos and Flach, 2002) (HBN) and Relational Bayesian Networks (Koller and Pfeffer, 1998) (RBN) which are in the core of the current thesis. RBNs are an extension of Bayesian networks in the relational context. As the relational data representation is widely different from the propositional data representation assumed by BNs, we devote the next chapter to present some useful concepts from the relational database theory.

2

Database Relational Model

The relational representation is strictly more expressive than propositional representation assumed by almost all machine learning techniques such as Bayesian Networks.
The relational model principles where originally laid down by E. F. Codd (Codd, 1983) who realized that the discipline of mathematics could be used to inject some solid principles and rigor into the database management field. Based on set theory and predicate logic, the relational model presents the theoretical foundation underlying today's relational products.
Nowadays, the relational model is the most commonly used database model, it represents the basis for the most large scale knowledge representation systems.
Systems used to handle relational data representation are called relational database management systems (RDBMS). Usually, these softwares have to be tested in case of update (e.g., new components added, old ones updated). Leading database platform providers (e.g., Microsoft, Oracle, Informix) often perform benchmark data to prove the performance of their products. Benchmarking database systems presents an old issue for the database community which is concerned with several axes, among which, determining data distribution and generating the database.
The chapter is dedicated to provide some useful concepts. Then, to give a formal definition of the relational model. Relational database benchmarking and methods to randomly generate databases are also discussed.

Contents

2.1 Introduction

A database system is usually classified based on the data structure and operators used to access the data. The relational representation has turned out sufficient for almost any kind of data representation problem, and most commercial databases are based on the basis of the relational model.

This situation gives rise to the idea of setting up a standard for measuring the performance of competitive products. The standard is usually known as a benchmark, A method for measuring similar systems in reference to a standard is called benchmarking. Using these benchmark measurements, it will be possible to compare the performance of various relational systems and to analyze their strengths and weaknesses

The remainder of this chapter is as follows: Section 2.2 gives an overview of database management. Section 2.3 presents some basic concepts from the relational representation and defines the relational model. Section 2.4 presents mapping rules from the Entity-relationship diagram to the relational model. Section 2.5 goes through relational database benchmarking and random relational database generation.

2.2 Database management

A database system is a repository for a collection of computerized files whose overall purpose is to maintain information and to make that information available on demand. It consists on four components, namely data, hardware, software and users (Date, 2003).

- **Data.** The data stored in the system. It can be shared or integrated.
- **Hardware.** Storage volumes used to store the data as well as processors and associated memory used to support the execution of the database system software.
- **Software.** All requests from users for access to the database are handled by the database management system (DBMS). It provides users with a view of the database and supports user operations (e.g., inserting, retrieving, updating data)
- **Users.** Users can be divided into three categories:
  - *Application programmers:* who responsible for writing application programs that use the database.
  - *End users:* who interact with the system from online terminals.
  - *Database administrator (DBA):* who is responsible for administrating the database and the database system.

One of the most important benefits of a database system is data independence, that is, the immunity of applications to changes in the way the data is stored and accessed.

Database systems are categorized based on the data structure and operators used to access the data. For instance, in the hierarchic system, the data is presented in the form of a set of trees structures and operators provided for manipulating such structures include operators for traversing hierarchic paths. In the relational system, the data is perceived by users as tables and the operators at the user's disposal generate new tables from old ones. Form both an economic and a theoretical perspective, the relational approach is easily the most important.The next section is dedicated to an extended explanation of the relational model components.

2.3 Relational model

A relational system is a system that supports relational databases and operations on such databases. This section starts by providing some basic concepts. Then we give the definition of a relational model. All these concepts and definitions are derived from (Date, 2003; Date, 2005; Date, 2008).

2.3.1 Basic concepts

The formal theory underlying relational systems is called the relational model. It addresses three aspects of data: data structure, data integrity and data manipulation.

2.3.1.1 Data structure

Definition 2.3.1. ***Domain.***
A domain D is a data type. It is a named set of scalar values $\mathcal{V} = (v_1, \ldots, v_l)$. It might be either predefined or user-defined. The later has a name that is unique within the database.

Definition 2.3.2. ***Attribute.***
Loosely, a column; more precisely, an $< attribute - name : domain - name >$ pair, though it's common to refer to a given attribute informally by its attribute name alone. Attributes have names that are unique within the containing relation.

Definition 2.3.3. ***Relation.***
A relation R consists of two parts: a heading and a body. The heading is the row of column headings and the body is the set of data rows.

- *The heading (= relation schema) consists of a fixed set of attributes, or more precisely $\langle attribute - name : domain - name \rangle$ pairs $(A_1 : D_1, \ldots, \langle A_n : D_n \rangle)$, such that each attribute A_j corresponds to exactly one of the underlying domains $D_j (j = 1, \ldots, n)$. The attribute names $A_1, \ldots, A_n$ are all distinct.*
- *The body consists of a set of tuples. Each tuple consists of a set of $\langle attribute - name : attribute - value \rangle$ pairs $(A_1 : Vvi_1, \ldots, \langle A_n : vi_n \rangle)$, where $(i = 1, \ldots, m)$ and m is the number of tuples in the set. In each such tuple, there is one such $\langle attribute - name : attribute - value \rangle$ pair $\langle A_j, vi_j \rangle$ for each attribute A_j in the heading. For any given pair $\langle A_j, vi_j \rangle$, vi_j is a value from the unique domain D_j that is associated with attribute A_j.*

The number of attributes, n, is called the degree and the number of tuples, m, is called the cardinality. Relations have names that are unique within the database.

Example 2.3.1. *Figure 2.1 summarizes all the concepts related to a relation. The relation heading is defined through 4 attributes, each of which has its one domain. The relation body is a set of m tuples. The relation degree is equal to 4 and its cardinality is equal to m.*

All relations satisfy four very important properties (Date, 2003).

1. They do not contain any duplicate tuples: This property follows from the fact that the body of the relation is a mathematical set, and sets in mathematics by definition do not include duplicate elements.
2. There is no ordering to the tuples: This property also follows from the fact that the body of the relation is a mathematical set and sets in mathematics are not ordered.
3. There is no ordering to the attributes: This property follows from the fact that the heading of a relation is also defined as a set of attributes.
4. All attributes values are atomic: This property is a consequence of the fact that all underlying domains contain atomic values only. A relation satisfying this condition is said to be **normalized**.

Definition 2.3.4. ***Relational schema.***
Sometimes referred to as a database schema, it is a collection of named relation schemas (relation headings).

Definition 2.3.5. ***Relational database.***
A relational database is a database that is perceived by its users as a collection of normalized relations (headings and body) of assorted degrees.

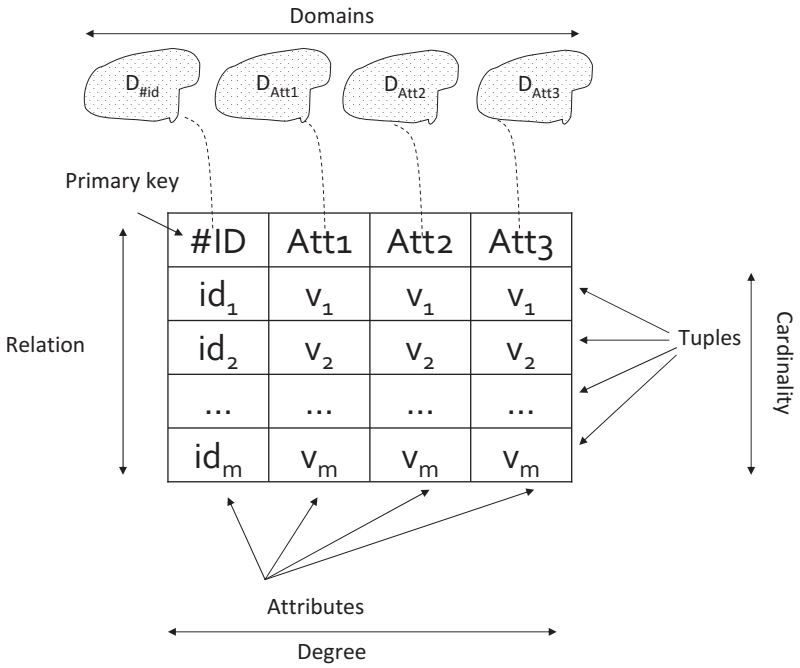


Figure 2.1: Relation components

2.3.1.2 Data integrity

Data integrity is insured by the use of primary and foreign keys.

Definition 2.3.6. ***Primary key.***
It is a unique identifier. More precisely, let K be a subset of the heading of a relation R. K is a primary key for R if and only if:

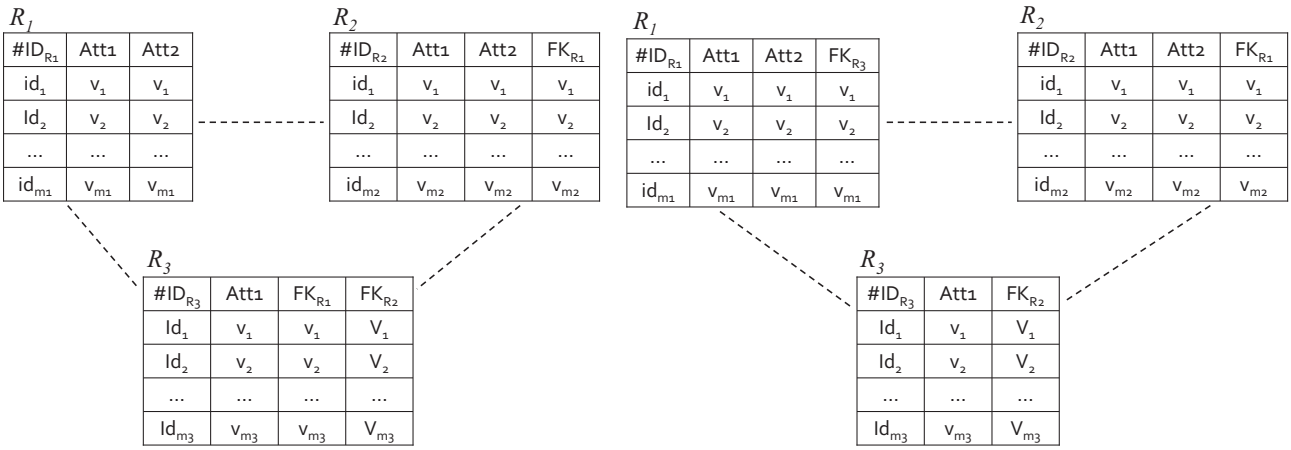
- *No possible value for R contains two distinct tuples with the same value for K (the uniqueness property);*
- *The same can't be said for any proper subset of K (the irreducibility property). Every relation R, has at least one candidate key. They are sets of attributes (and key values are therefore tuples), namely, a column or column combination with the property that, at any given time, no two rows of the relation contain the same value in that column or column combination.*

Definition 2.3.7. ***Foreign key.***
Let R_1 and R_2 be relations, not necessarily distinct, and let K be a key for R_1. Let FK be a subset of the heading of R_2 such that there exists a possibly empty sequence of attribute renaming on R_1 that maps K into K' (say), where K' and FK contain exactly the same attributes. Further, let R_2 and R_1 be subject to the constraint that, at all times, every tuple $t2$ in R_2 has an FK value that's the K' value for some (necessarily unique) tuple $t1$ in R_1 at the time in question. Then FK is a foreign key, the associated constraint is a ***referential constraint****, and R_2 and R_1 are the referencing relation and the corresponding referenced relation, respectively, for that constraint.*

Along with the foreign key concept, the relational model includes the following rule:

Property 2.3.1. ***Referential integrity.***
The database must not contain any unmatched foreign key values. means that if B references A, then A must exist.

From the foreign key definition, we can introduce referential path and referential cycle notions.



(a) An example of a referential path (b) An example of a referential cycle

Figure 2.2: An illustration of foreign key, referential path, and referential cycle

Definition 2.3.8. ***Referential path.***
Let relations $R_z, R_y, R_x, ..., R_b, R_a$ be such that there exists a referential constraint from R_z to R_y, a referential constraint from R_y to R_x, . . . , and a referential constraint from R_b to R_a. Then the chain of such constraints from R_z to R_a constitutes a referential path from R_z to R_a (and the number of constraints in the chain is the length of the path).

Definition 2.3.9. ***Referential cycle.***
A referential cycle is a referential path from some relation to itself.

Note that database designs involving referential cycles are usually contraindicated (Date, 2008).

Example 2.3.2. *Let R_1, R_2 and R_3 be three relations, with primary keys $\#ID_{R1}$, $\#ID_{R2}$ and $\#ID_{R3}$ respectively.*

In Figure 2.2(a), R_2 references R_1 using the foreign key FK_{R1}. For each tuple t of relation R_2 has an FK_{R1} that exists in column $\#ID_{R1}$ of relation R_1. R_1 is the referenced relation. Moreover, there exists referential constraints from R_3 to R_2 and from R_2 to R_1. Then, $\{R_3, R_2, R_1\}$ constitutes a relational path of length 2.

In Figure 2.2(b), there exists a referential constraints from R_3 to R_2, from R_2 to R_1 and from R_1 to R_3. Then, $\{R_3, R_2, R_1, R_3\}$ constitutes a relational cycle.

2.3.1.3 Data manipulation

Data manipulation concerns operators used to access the data. They are all set-level operators. An operator manipulates individual data items and returns a result. The data items are called operands or arguments.

Property 2.3.2. ***The closure property.***
The closure property states that the output from every operation is the same kind of objects as the input (i.e., relations) which implies that we can write nested relational expressions.

Usually, the interaction with a relational database is ensured by specifying queries using structured language. The standard language for interacting with relational databases is the Structured Query Language (SQL) (Kline et al., 2008).

Based on the relational algebra and tuple relational calculus, SQL consists of a data definition language and a data manipulation language. It allows schema creation and modification, data update and delete, as well as data access control. It presents a standard of the American National Standards Institute (ANSI) and of the International Organization for Standardization (ISO).

SQL uses some specific operators to extract significant meaning such as aggregators and multi-set operators.

Definition 2.3.10. ***Aggregate function.***
An aggregate function γ takes a multi-set of values of some ground type, and returns a summary of it.

Multi-set operators combine the results of two queries into a single result (Perlich and Provost, 2006). Formally, a multi-set operation is defined as follows:

Definition 2.3.11. ***Multi-set operator.***
A multi-set operator ϕ_k on k multi-valued attributes $A_1, \ldots, A_k$ that share the same range $V(A_1)$ denotes a function from $V(A_1)^k$ to $V(A_1)$.

Example 2.3.3. *Example of aggregate functions could be the maximum, the minimum, the mode, the average, the number of rows, etc. While, the intersection and the union are examples of multi-set operators.*

For instance, SELECT ***COUNT*** *(*) FROM R_1; is an SQL query that returns the number of tuples in R_1.*

The SQL query SELECT $\#ID_{R1}$ FROM R_1 ***UNION*** *SELECT FK_{R1} from R_2; returns all $\#ID_{R1}$ that appeared in both R_1 and R_2.*

2.3.2 Relational model definition

The relational model represents a database system at a level of abstraction that is somewhat removed from the details of the underlying machine. Having introduced all basic concepts on which a relational model is found, we can formally define it as follows:

Definition 2.3.12. ***Relational model.***
A relational model addresses three aspects of data:
- *Data structure: a collection of relations.*
- *Data integrity: insured by the use of primary and foreign keys*
- *Data manipulation: operators used to access the data.*

2.3.3 Relational model representation

During this dissertation, we will adopt the following graphical representation to illustrate relational models.
- Rectangle to represent relations.
- Each rectangle has a head part representing the name of the entity and a body part representing the relation attributes.
- Each primary key is underlined.
- Each foreign key has the same name as the primary key attribute of the referenced relation.
- Dotted lines to represent referential constraints.

Example 2.3.4. *Figure 2.3 is an example of a relational model representation relative to 4 relations $Professor$, $Student$, $Course$ and $Registration$ of degrees 3, 3, 4 and 5 respectively. It contains 3 referential constraints: The relation $Course$ references the relation $Professor$. The relation $Registration$ references both relations $Student$ and $Course$.*

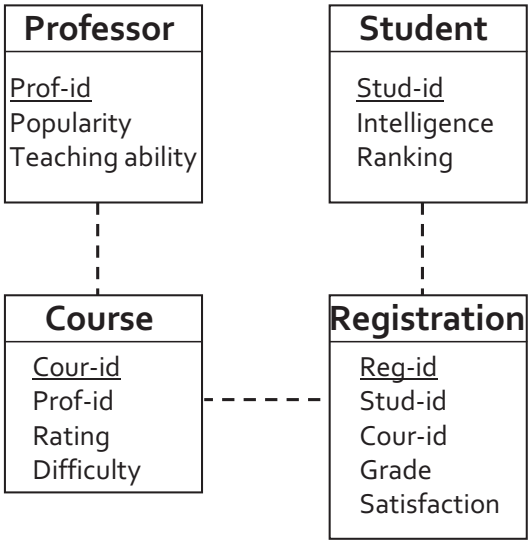


Figure 2.3: An example of a relational model representation

2.4 From the entity-relationship model to the relational model

The Entity-Relationship (ER) model provides the conceptual level of database design. It was initially defined by Bachman (Bachman, 1969) and Chen (Chen, 1976). An ER diagram can be easily mapped into a relational model to represent database schema. its strength lies on its basic building block simple and easy to understand [1]. It is based on two simple concepts, namely, entities and relationships (Sumathi and Esakkirajan, 2007).

2.4.1 Entities

An *entity* is a particular unique object. For instance, a particular person, a particular place, etc. A collection of similar entities forms the *entity type*. For instance, all students at the university are of type 'Student'. Each entity type has a set of properties that allow to describe entities. These properties are called *attributes*. For instance, enrollment number and name are attributes of the entity Student. Properties that might be unique within the context of an entity are called *keys*. For instance, The enrollment number is unique within the context of a particular student.

2.4.2 Relationships

Relationships are associations between one or more entities. They are classified into four main types, namely 1) One-to-Many relation 2) Many-to-One, 3) One-to-One relation and 4) Many-to-Many relation.

- **One-to-Many relationship type.** It associates one entity to more than one entity. For instance an order may contain many items. while an item belongs to only one order.
- **Many-to-One relationship type.** It associates more than one entity to just one entity. as relationships work both ways, the same example of Order and Item presents an example of Many-to-One relationship: There may be many items in a particular order.
- **One-to-One relationship type.** This relationship type is rarely met. It associates one entity to exactly one entity, and it is considered as a special case of One-to-Many relationship. As an example of One-to-One relationship is the relationship between a person and a passport in a particular country. A person has only one passport, a particular passport is delivered to only one person.

1. In this part we are limited to only useful definition that we will reuse during the rest of the manuscript. For further reading and more detailed definitions, we refer readers to (Date, 1990; Date, 2003; Sumathi and Esakkirajan, 2007)

Relationship type	Representation
One-to-One	
One-to-Many	
Many-to-One	
Many-to-Many	

Table 2.1: Relationship types and representations

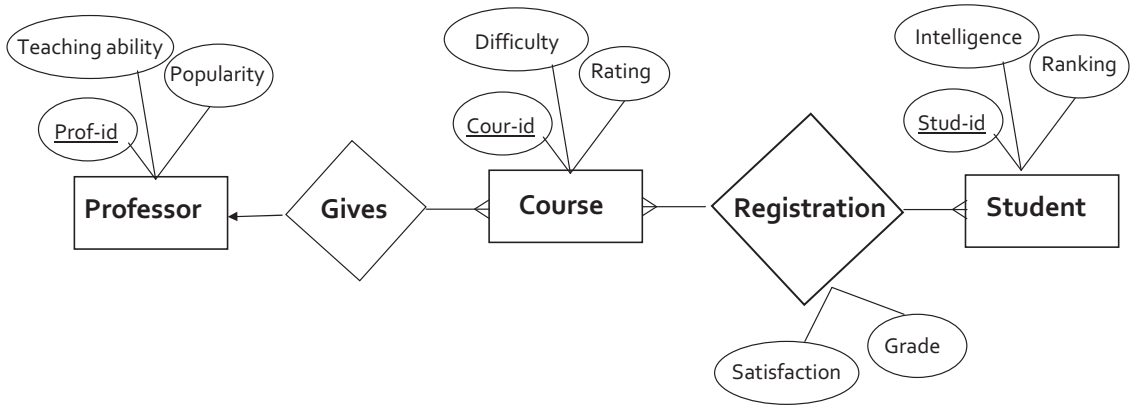


Figure 2.4: An example of an Entity-relationship diagram

– **Many-to-Many Relationship Type.** It associates more than one entity to more than one entity. For instance A student can take many courses and a course can have many students.

2.4.3 Entity-relationship diagram representation

Following (Sumathi and Esakkirajan, 2007), an ER diagram is represented by:
– Rectangle to represent entity types.
– Ellipses to represent attributes of entity types.
– Diamonds to represents relationships.
– Lines to represent linking of attributes to entity types and of entity types to relationship types.
– The four relationship types are summarized in Table 2.1

Example 2.4.1. *Figure 2.4 is an example of an entity-relationship diagram. The example contains three entity types, namely, Professor, Course and Student. Two relationships: a One-to-Many relationship between Professor and Course and a Many-to-Many relationship between Course and Student. The Professor attributes are* $Prof - id$ *as identifier, Popularity and Teaching ability.*

2.4.4 Mapping ER Diagram to a relational model

An ER diagram can be easily mapped to a relational model by applying a set of mapping rules that we list bellow (Sumathi and Esakkirajan, 2007):
– **Rule 1.** To each entity type E corresponds a relation (table). The columns of the table are the attribute of the entity type E.

- **Rule 2.** *Mapping One-to-Many (Many-to-One) Relationship:* Create a relation for each entity type participating to the relationship. Then, include the primary key attribute of the entity on the One-side of the relationship as a foreign key in the relation that is on the Many-side of the relationship.
- **Rule 3.** *Mapping one-to-one relationship:* As One-to-One relationship is a particular case of One-to-Many relationship. The mapping processes are similar: First, two relations are created, one for each of the entity types. Then, the primary key of either sides is included as a foreign key to the other side. Another possible mapping of One-to-One relationships becomes to subsume one entity type into the other (Date, 1990).
- **Rule 4.** *Mapping Many-to-Many Relationship:* The relationship is transformed to an entity type. Then three relations have to be created: one for each of the three entity types. Primary key attributes of the two initial entities are included as identifiers in the relation that presents the relationship. These keys can be used together as a primary for the third relation, also, a new identifier can be declared.
- **Rule 5.** All attributes of the relationship, if any, go to the side receiving the transferred primary key.

Example 2.4.2. *Figure 2.3 presents the mapping of the ER diagram of Figure 2.4 using the previously listed rules.*

2.5 Benchmarking database systems

Long ago, and due to the absence of 'standard' to measure a system performance against other systems, each DBMS vendor (e.g., Microsoft, Oracle, Informix) has its own means to evaluate its product. It cites benchmark data that proves its database is the fastest, and each vendor chooses the benchmark test that presents its product in the most favorable light. Often results are not divulged. If they are leaked, vendors who present lower results try usually to discredit the benchmark. This phenomenon is known as *benchmark wars*.

Consequently, some effort has been made in order to provide uniform benchmark tests and deliver trusted results to the industry. Due to the importance of this task, the Transaction Processing Performance Council (TPC) organization has been founded to define transaction processing and database benchmarks to be disseminated to the industry [2]. The TPC recognizes its benchmarks with respect to application types and provides ways to calculate system price and report performance results. Till now, their benchmarks present a standard to utmost all database system vendors.

From there, other commercial and non commercial benchmarks have been instituted. Examples include Wisconsin (DeWitt, 1993), Engineering Database Benchmark (Cattell, 1993) and Neal Nelson's Business Benchmark [3].

Database benchmarks still remains trendy (Curino et al., 2012; Angles et al., 2013; Difallah et al., 2013) because of hardware and software development and also the huge amount of data provided due to the development of communication technologies and the need of storage it and manage it in such a manner it can be used later.

In this section, we provide database benchmarks definition, types and components. Then we focus on a particular element of a database benchmark, namely the database generation. Finally we discuss a particular family of database benchmarks for decision support systems.

2.5.1 Database Benchmarks definition

A database benchmark should have some important basis such as (Gray, 1993):
- Generating the transaction workload.
- Defining transaction logic.
- Defining data access requirements.

2. http://www.tpc.org
3. http://www.nna.com/

– Determining data distributions and generating the database.
– Instrumenting and measuring the system.

2.5.1.1 Database benchmark types

Generally, database benchmarks are used to measure the performance of a database system. However, it is not its only objective. In fact, we distinguish two types of database benchmarks:

– **Performance benchmarks.** allow to measure response time (e.g., transactions per second), batch throughput and current hardware capacities.
– **Functional benchmarks.** allow to measure the system functionality, ease of use, operability, ease of development, as well as other aspects of data processing.

Database benchmarks can be classified alternatively on generic benchmarks and domain-specific ones. The former are useful in comparing systems overall performance. The latter allow to measure the performance with respect to the application domain. Gray (Gray, 1993) reports that such a benchmark has to be relevant, portable, scalable, and simple.

2.5.1.2 Database benchmark components

A database benchmark has two major components (Gray, 1993):

– **The workload specification.** It requires some analysis of what we hope to implement: If the system is completely new, some effort must go into database and application design before any work specification can be performed. Otherwise, extension has to be made while respecting the current system specification.
– **The measurement specification.** The major decision is *how much of the environment will be measured?* Trade-off between reality and simplicity should be taken into account when defining the benchmark specification.

2.5.2 Random relational database generation

Database generation is a particular element of a database benchmark. It presents an old issue for database designers. Its interest occurs when evaluating the effectiveness of a RDBMS components. Consequently, several propositions have been developed in this context. The main issue was how to provide a large number of records using some known distributions in order to be able to evaluate the system results. (Bitton et al., 1983; Bruno and Chaudhuri, 2005). In some research, known benchmarks [4] are used and the ultimate goal is only to generate a large dataset (Gray et al., 1994). Several software tools are available (e.g. *DbSchema* [5], *DataFiller* [6]). They allow to populate database instances knowing the relational schema structure. Records are then generated on the basis of this input.

Nowadays, the widespread use of Internet leads to the generation of huge and complex amount of data. The challenges of capturing, storing, and analyzing, such big data imply further development of big data systems, which in return implies the development of big data benchmarks in order to evaluate and compare such systems. For such benchmarks, it is not obvious to obtain real big data sets to use as workload inputs especially for confidential issues. In addition, even if such a dataset can be provided, it will not be sufficient to satisfy all benchmarking scenarios (Lu et al., 2014; Zhu et al., 2014).

Synthetic data generation is a natural solution to solve this issue and generally a degree of predictability in the generated data is necessary, especially for domain-specific benchmarks, where veracity of the real life data has to be taken into account while generating data. For instance, in TPC-D benchmark, attribute values are uniformly distributed (Ballinger, 1993).

4. http://www.tpc.org
5. http://www.dbschema.com/
6. https://www.cri.ensmp.fr/people/coelho/datafiller.html

2.5.3 Benchmarking for decision support systems

There are several families of Database benchmarks (Darmont, 2009) and some of them have been conceived for specific applications (e.g, decision support [7], Cloud Computing (Ferrarons et al., 2013)). Great effort has been devoted by the TPC in order to provide DB benchmarks for decision support applications. Early work led to the creation of the TPC-D benchmark [8]. This latter has been substituted by the TPC-R [9] benchmark in 1999. From 2005, TPC-R became an obsolete benchmark and has been replaced by the TPC-DS [10] considered, till now, as the decision support benchmark standard. TPC-DS is a benchmark for data warehouses and decision support systems. It has been designed to be suitable with real-world business tasks which are characterized by the analysis of huge amount of data. Its schema models the sales and sales returns process for an organization. TPC-DS provides tools to generate either the data sets or the query sets for the benchmark. Nevertheless, uncertainty management stays a prominent challenge to provide better rational decision making.

2.6 Conclusion

In this chapter we have defined the relational model. Then we presented relational database benchmarking and we have focused especially on random database generation process. We have shown that benchmarking has been addressed only from the database industry perspective, where the main objective is to provide common frameworks to compare the performance of different relational systems.

After presenting the BN model in Chapter 1 and the relational representation in the current chapter, Chapter 3 introduces an extension of Bayesian networks in the context of relational data representation, namely, Relational Bayesian Networks.

7. http://www.tpc.org/tpcds
8. http://www.tpc.org/tpcd
9. http://www.tpc.org/tpcr
10. http://www.tpc.org/tpcds

3

Relational Bayesian Networks

During the last decade, there has been growing interest in extracting patterns from relational data representation. Statistical relational learning (SRL) is an emerging area of machine learning that enable effective and robust reasoning about relational data structures (Heckerman et al., 2007). Relational Bayesian networks (RBNs) are an extension of Bayesian networks which allow to work with relational database representation rather than propositional data representation. Initially, this model was called probabilistic relational model (PRM) (Koller and Pfeffer, 1998; Pfeffer, 2000). Then, Neville and Jensen (Neville and Jensen, 2007) proposed to preserve the use of the term PRM in its more general sense to distinguish the family of PGMs that are interested in extracting statistical patterns from relational models, and to use RBN to refer to Bayesian networks that have been upgraded to model relational databases. In our dissertation, we preserve the same designation as (Neville and Jensen, 2007). Note that (Koller and Pfeffer, 1998; Pfeffer, 2000) model is different from the relational Bayesian network of Jaeger (Jaeger, 1997) that is an extension of BN using the first-Order logic. RBNs are interested in producing and manipulating structured representations of the data, involving objects described by attributes and participating in relationships, actions, and events. The probability model specification concerns classes of objects rather than simple attributes.

RBNs have demonstrate their applicability in several areas (e.g., industry, system quality analysis, web page classification, etc.) (Fersini et al., 2009; Sommestad et al., 2010) and till now works are in progress in order to provide solid theoretical foundations and develop appropriate algorithms allowing their construction from relational data. RBNs learning is inspired from standard BNs learning approaches and naturally their evaluation will be similar to BNs evaluation techniques but unfortunately this issue has not been well tackled. The chapter is then dedicated to introduce this framework and to make a survey on existing learning as well as evaluating approaches.

Contents

3.1 Introduction

Relational Bayesian networks (RBNs) present an extension of BNs in the relational context. In this chapter we will focus on RBNs defined by (Koller and Pfeffer, 1998; Pfeffer, 2000). The goal behind this formalism is to express probabilistic models in a compact and intuitive way that reflects the relational structure of the domain and, ideally, supports efficient learning and inference (Heckerman et al., 2007). As for standard BNs, RBNs construction implies the identification of the graphical structure and parameters. RBNs learning from relational data has been inspired from classical BNs learning approaches. An intuitive way to evaluate these learning approaches is the adaptation of evaluation techniques used for BNs to the relational context.

The remainder of this chapter is as follows: Section 3.2 defines relational Bayesian networks and similar model representations. Section 3.3 crosses RBNs learning approaches. Section 3.4 goes through evaluation techniques of RBNs learning approaches. Section 3.5 presents a list of some existing relational Bayesian network-related softwares.

3.2 Relational Bayesian network formalism

In this section, we give a formal definition of relational Bayesian networks. Then, we discuss RBNs acyclicity and relational d-separation. Finally, we briefly present structural uncertainty in RBNs. Recall that we focus on RBNs of (Koller and Pfeffer, 1998; Pfeffer, 2000) in this thesis work. Jaeger (Jaeger, 1997) used the same terminology to refer to another model specification which extends BNs using the first-Order logic.

3.2.1 Relational Bayesian network definition

Let $\mathcal{R}$ be a relational schema, $\mathcal{X}$ be the set of all classes of a RBN, $\mathcal{A}(X)$ be the set of descriptive attributes of a class $X \in \mathcal{X}$ and $\mathcal{R}(X)$ be the set of reference slots of a class X.

Definition 3.2.1. ***Reference slot.*** *A reference slot of class X denoted by $X.\rho$ has X as domain type and Y as a range type, where $Y \in \mathcal{X}$. For each reference slot ρ, a reversed slot ρ^{-1} can be defined.*

Definition 3.2.2. ***Slot chain.*** *A slot chain K is a sequence of reference slots. (reversed or not) $\rho_1, \ldots, \rho_k$, where $\forall i, Range[\rho_i] = Dom[\rho_{i+1}]$.*

A RBN Π for a relational schema $\mathcal{R}$ is defined through a qualitative dependency structure $\mathcal{S}$ and a set of parameters associated with it $\theta_{\mathcal{S}}$.

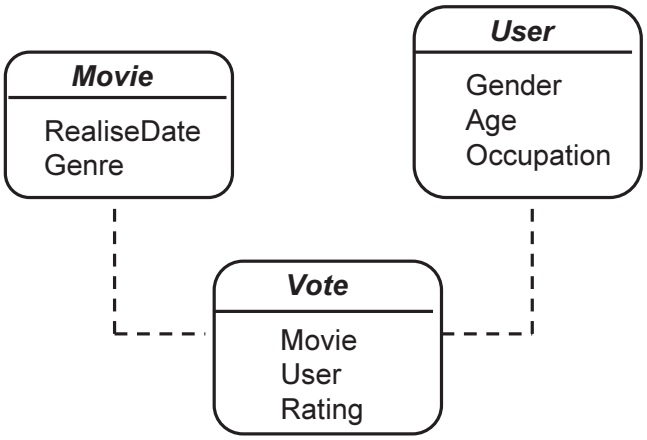
Definition 3.2.3. ***Relational Bayesian network.***

Formally, a RBN Π for a relational schema $\mathcal{R}$ is defined by (Getoor et al., 2007):

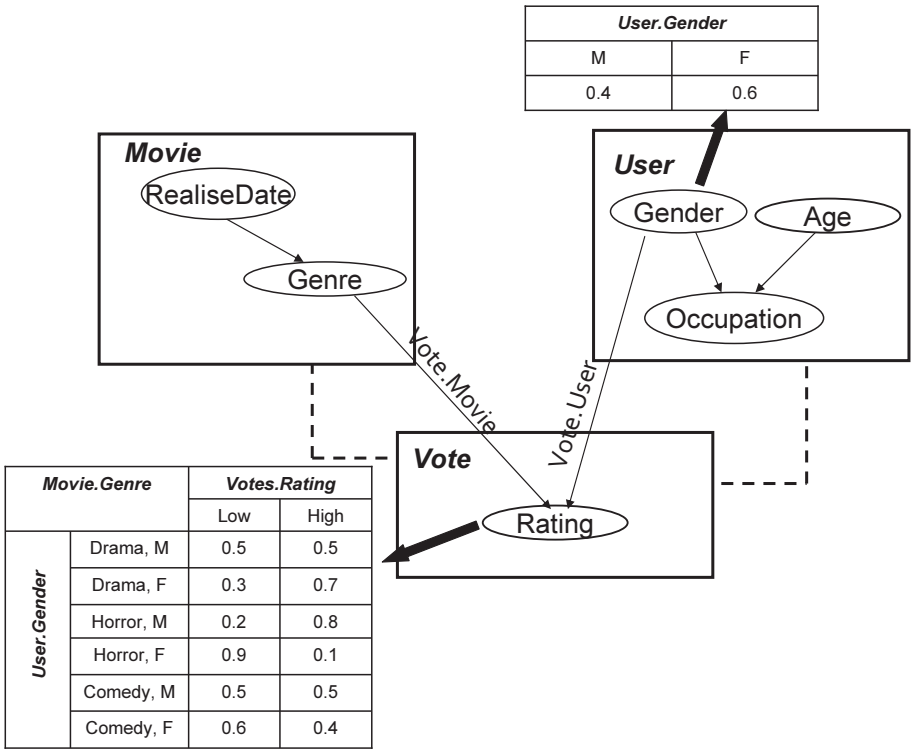
- *A qualitative dependency structure $\mathcal{S}$: For each class (relation) $X \in \mathcal{X}$ and each descriptive attribute $A \in \mathcal{A}(X)$, there is a set of parents $Pa(X.A) = \{U_1, \ldots, U_l\}$ that describes probabilistic dependencies. Each U_i has the form $X.B$ if it is a simple attribute in the same relation or $\gamma(X.K.B)$, where K is a slot chain and γ is an aggregate of $X.K.B$ (cf. Section 2.3.1).*
- *A set of conditional probability distributions (CPDs), representing $P(X.A|Pa(X.A))$.*

Note that there is a direct mapping from RBNs representation to relational database concepts: A class correspond to a single relation (or table). Descriptive attributes correspond to standard table attributes. Reference slots correspond to foreign key attributes. Slot chains correspond to long reference paths, with some possible back and forth.

Slot chains may be arbitrary large and give rise to complicated models. So that a user specified value, the maximum slot chain length (K_{max}), is required to limit the length of possible slot chains that one can cross in the model.



(a) An example of a relational schema



Movie.Genre		Votes.Rating	
		Low	High
User.Gender	Drama, M	0.5	0.5
	Drama, F	0.3	0.7
	Horror, M	0.2	0.8
	Horror, F	0.9	0.1
	Comedy, M	0.5	0.5
	Comedy, F	0.6	0.4

(b) An example of a RBN

Figure 3.1: An example of a relational schema and a RBN for a movie domain

Example 3.2.1. *An example of a relational schema is depicted in Figure 3.1(a), with three classes* $\mathcal{X} = \{Movie, Vote, User\}$. *The class* $Vote$ *has a descriptive attribute* $Vote.Rating$ *and two reference slots* $Vote.User$ *and* $Vote.Movie$. $Vote.User$ *relates the objects of class* $Vote$ *with the objects of class* $User$. *Dotted links presents reference slots. An example of a slot chain would be* $Vote.User.User^{-1}.Movie$ *which could be interpreted as all the votes of movies shown by a particular user.*

$Vote.Movie.genre \rightarrow Vote.rating$ *is an example of a probabilistic dependency derived from a slot chain of length* 1 *where* $Vote.Movie.genre$ *is the parent and* $Vote.rating$ *is the child as shown by Figure 3.1(b). Also, varying the slot chain length may give rise to other dependencies. For instance, using a slot chain of length* 3*, we can have a probabilistic dependency from* $\gamma(Vote.User.User^{-1}.Movie.genre)$ *to* $Vote.rating$. *In this case,* $Vote.rating$ *depends probabilistically on an aggregate value of all the genres of movies rated by a particular user.*

Roughly speaking, a RBN Π is a meta-model used to describe the overall behavior of a system, to perform probabilistic inference this model, as well as the relational schema have to be instantiated.

An instance of a relational schema is defined as follows (Getoor et al., 2007):

Definition 3.2.4. ***Instance of a relational schema.*** *An instance $\mathcal{I}$ of a schema specifies:*
- *for each class X, the set of objects in the class, $\mathcal{I}(X)$.*
- *a value for each attribute $x.A$ (in the appropriate domain) for each object x.*
- *a value y for each reference slot $x.\rho$, which is an object in the appropriate range type, i.e., $y \in Range[\rho]$. Conversely, $y.\rho^{-1} = x|x.\rho = y$.*

A RBN instance contains for each class of Π the set of objects involved by the system and relations that hold between them (i.e., tuples from the database instance which are interlinked). This structure is known as a relational skeleton σ_r (Getoor et al., 2007).

Definition 3.2.5. ***Relational skeleton.*** *A relational skeleton σ_r of a relational schema is a partial specification of an instance of the schema. It specifies the set of objects $\sigma_r(Xi)$ for each class and the relations that hold between the objects. However, it leaves the values of the attributes unspecified.*

Given a relational skeleton, the RBN Π defines a distribution over the possible worlds consistent with σ_r.

Definition 3.2.6. ***Ground Bayesian Network.*** *A RBN Π together with a relational skeleton σ_r define a ground Bayesian network (GBN) with:*
- *a qualitative component:*
  - *a node for every attribute of every object $x \in \sigma_r(X), x.A$.*
  - *each $x.A$ depends probabilistically on a set of parents $Pa(x.A) = \{u_1, \ldots, u_l\}$ of the form $x.B$ or $x.K.B$. If K is not single-valued, then the parent is the aggregate computed from the set of random variables $\{y|y \in x.K\}, \gamma(x.K.B)$.*
- *a quantitative component, the CPD for each $x.A$ is $P(X.A|Pa(X.A))$.*

Example 3.2.2. *The Figure 3.2(a) is a possible relational skeleton of the relational schema of Figure 3.1(a). It contains 3 users, 5 movies and 9 votes, where user 1 has voted for movies 1 and 2 and so on. A relational skeleton together with a RBN model define the ground Bayesian network of Figure 3.2(b) (CPDs of objects are those of the RBN, Figure 3.1(b)).*

As with standard Bayesian networks, the joint distribution over the instantiations compatible with our particular skeleton σ_r is factored, which leads to the following chain rule:

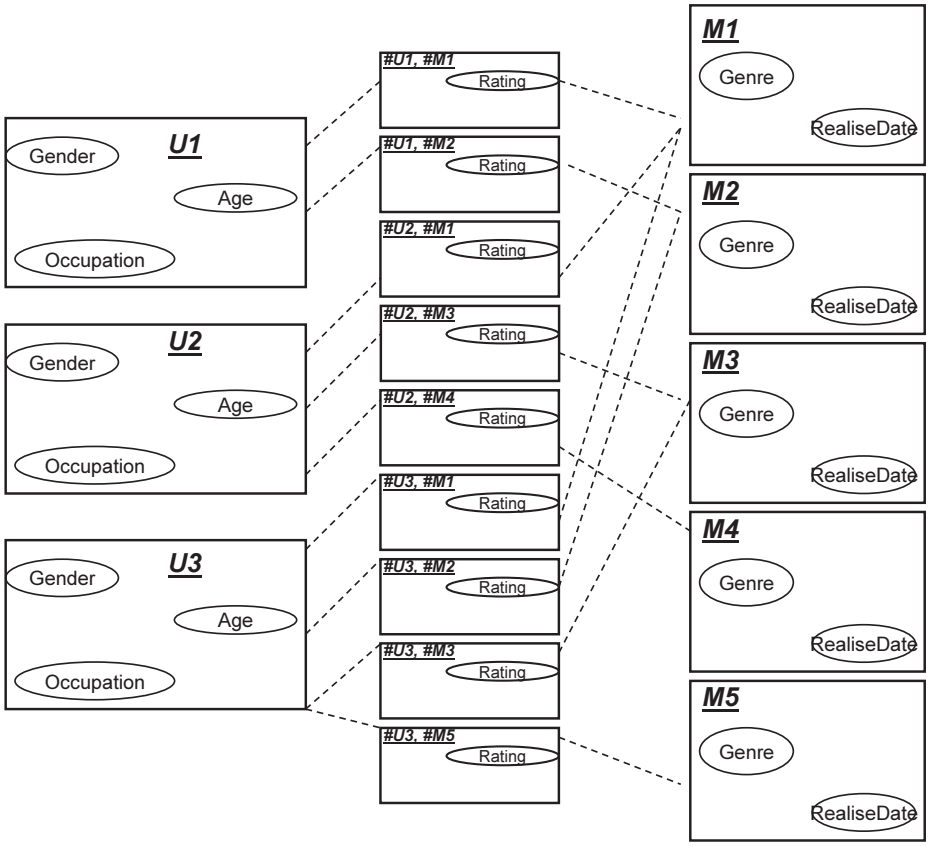
$$P(\mathcal{I}|\sigma_r, \mathcal{S}, \theta_{\mathcal{S}}) = \prod_{X \in \mathcal{X}} \prod_{x \in \sigma_r(X)} \prod_{A \in \mathcal{A}(X)} P(x.A|Pa(x.A)) \tag{3.1}$$

As for standard BNs, The dependency graph must define a coherent probability distribution. The next section presents some concepts (Getoor et al., 2007) that ensure coherence in RBNs.

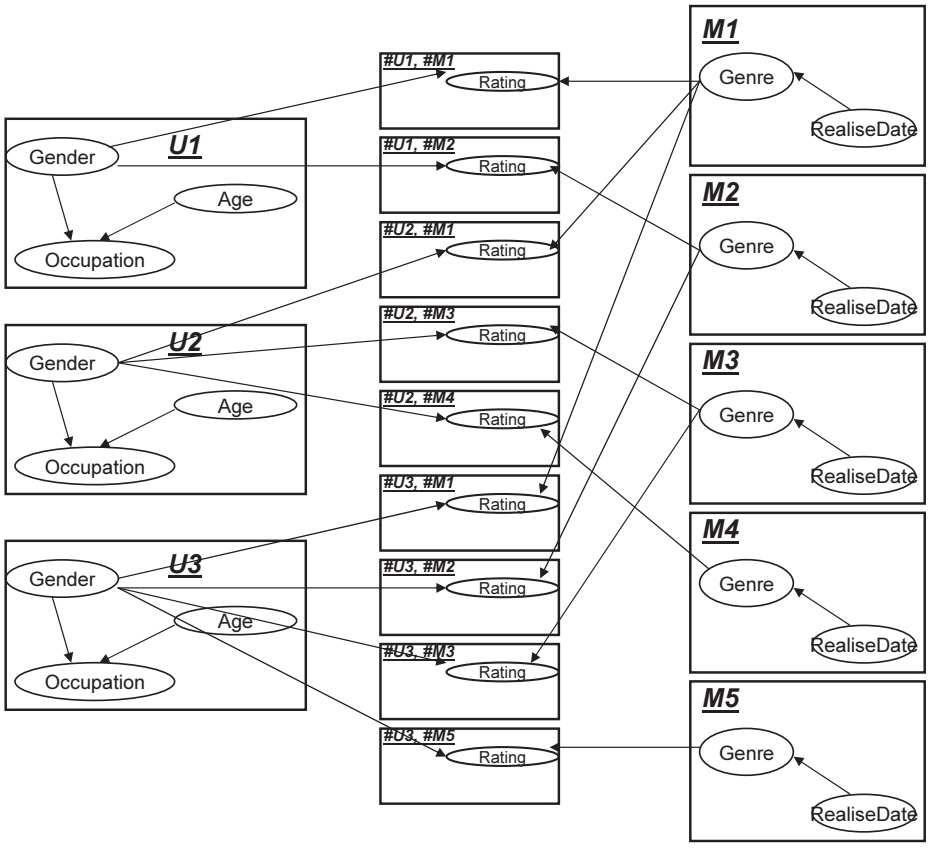
3.2.2 Cycles in relational Bayesian networks

Pfeffer (Pfeffer, 2000) introduces the *class dependency graph* notion that he uses to ensure that the probabilistic dependencies defined by the RBN are acyclic (i.e., a random variable does not depend, directly or indirectly, on its own value.).

Definition 3.2.7. ***Class dependency graph.*** *The class dependency graph $\mathcal{G}_\Pi$ for a RBN Π has as:*
- *Nodes: a node for each descriptive attribute $X.A$.*
- *Edges: Type I and Type II edges where:*

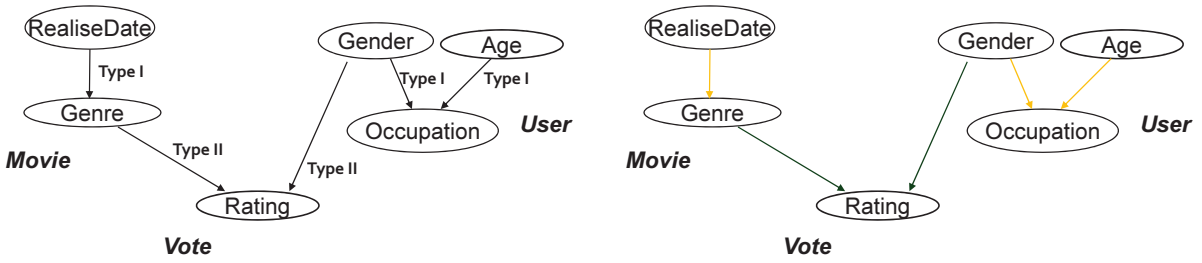


(a) An example of a relational skeleton



(b) An example of a GBN

Figure 3.2: An example of a relational skeleton and a GBN for a movie domain



(a) The class dependency graph for the RBN of Figure 3.1(b)

(b) The colored class dependency graph for the RBN of Figure 3.1(b)

Figure 3.3: Example of a class dependency graph and its corresponding colored class dependency graph

1. ***Type I edges:*** *For any attribute $X.A$ and any of its parents $X.B$, there is an edge from $X.B$ to $X.A$.*
2. ***Type II edges:*** *For any attribute $X.A$ and any of its parents $X.K.B$ there is an edge from $Y.B$ to $X.A$, where $Y = Range[X.K]$.*

In some situations, a cycle in the class dependency graph is not problematic, if it does not result in a cycle at the ground level. (e.g., a recursive model: genetic model of the inheritance of a single gene that determines a person's blood type (Getoor et al., 2007)). To represent such models using RBNs, Pfeffer (Pfeffer, 2000) defines the *colored class dependency graph* to guaranty the absence of cycles.

Definition 3.2.8. ***Colored class dependency graph.*** *The colored class dependency graph $\mathcal{G}_\Pi$ for a RBN Π has the following edges:*

- ***Yellow edges:*** *if $X.B$ is a parent of $X.A$, we have a yellow edge $X.B \rightarrow X.A$.*
- ***Green edges:*** *if $\gamma(X.K.B)$ is a parent of $X.A$, $Y = Range[X.K]$, and K is guaranteed acyclic, we have a green edge $Y.B \rightarrow X.A$.*
- ***Red edges:*** *if $\gamma(X.K.B)$ is a parent of $X.A$, $Y = Range[X.K]$, and K is not guaranteed acyclic, we have a red edge $Y.B \rightarrow X.A$.*

Definition 3.2.9. ***Stratified class dependency graph.*** *A colored class dependency graph is stratified if every cycle in the graph contains at least one green edge and no red edges.*

Example 3.2.3. *Figure 3.3(a) is an example of a graph dependency graph where probabilistic dependencies are labeled with respect to the edge type. The colored class dependency graph of Figure 3.3(b) contains 3 yellow edges representing intra-class dependencies of the form $X.B \rightarrow X.A$ and 2 green edges representing inter-class dependencies of the form $\gamma(X.K.B) \rightarrow X.A$. It is a stratified class dependency graph as it does not contain red edges.*

3.2.3 Related work

Even if pioneering work on establishing relational extension of directed graphical models has been found in (Koller and Pfeffer, 1998), other similar representations have been proposed in order to provide more clear theoretical basis and to generalize among all models of this family. Heckerman et al. (Heckerman et al., 2004) defined the Directed Acyclic Probabilistic Entity-Relationship (DAPER) model and have discussed that a DAPER model can be easily transformed into a RBN as both are probabilistic graphical languages for relational data and both follow a directed acyclic graphical representation. Maier et al. (Maier et al., 2013c) proposed formal definitions of relational domain concepts similar to those of RBN and DAPER. In this section, we start by giving a brief representation of DAPER models and we represent a mapping from DAPER to RBN. Then, we will especially focus on Maier et al. (Maier et al., 2013c) definitions as they will be useful to announce relational d-separation later (cf. Section 3.2.4).

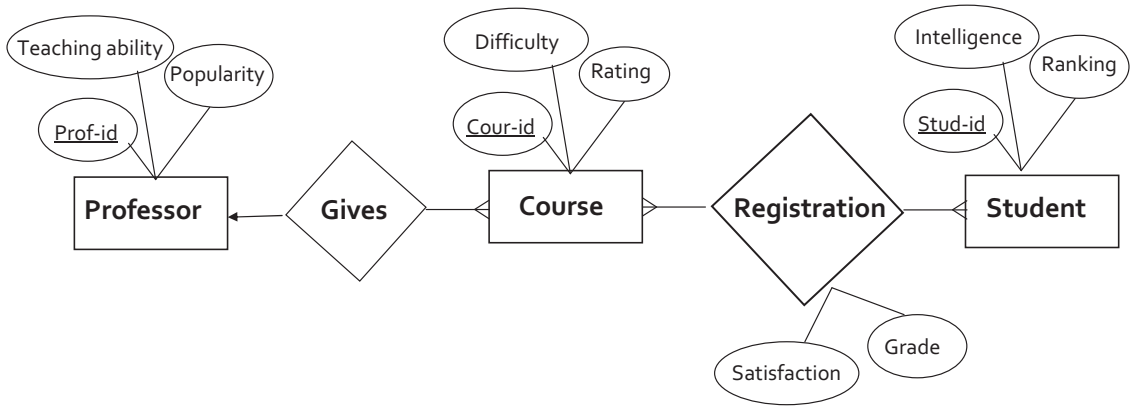


Figure 3.4: The entity-relationship representation of the university domain

3.2.3.1 The DAPER Model

DAPER models are to entity-relationship models (cf. Section 2.4) what RBNs are to relational schema (Heckerman et al., 2004).

Definition 3.2.10. ***Directed Acyclic Probabilistic Entity-Relationship model.*** *DAPER model is a directed extension of the entity-relationship model. It consists of two components:*

- *A qualitative component: The DAPER graph constructed among the ER model to which directed arcs between attribute classes are added to represent probabilistic dependencies*
- *A quantitative component: local distributions for attributes.*

Unlike RBN model, which uses aggregation functions to summarize a multi-set of values, DAPER refers to the local distributions of such dependencies as canonical distribution and does not describe how to encode them.

As for RBN, given a skeleton, the DAPER model allows to express conditional independence among realized attributes.

Heckerman et al. (Heckerman et al., 2004) have provided an invertible mapping from a DAPER model to a RBN. Table 3.1 summarizes the main steps of this mapping.

DAPER	RBN
ER model	Relational model
Probabilistic component of DAPER	Probabilistic component of RBN
Entity classes	Tables
Relationship classes	Tables with foreign keys making the connections to to tables presenting the entities
Attribute classes	Attributes
Arc classes and constraints	Drawn as they are in DAPER model

Table 3.1: Mapping rules from DAPER to RBN (Heckerman et al., 2004)

The final RBN resulting from those mapping rules has its probabilistic component defined as a graphical augmentation of the relational model. While in a RBN, the probabilistic component takes the form of a list of arc classes. Figure 3.4 provides the entity-relationship representation of the university domain and Figure 3.5 presents the relational schema resulting from this ER diagram with respect to the DAPER formalism.

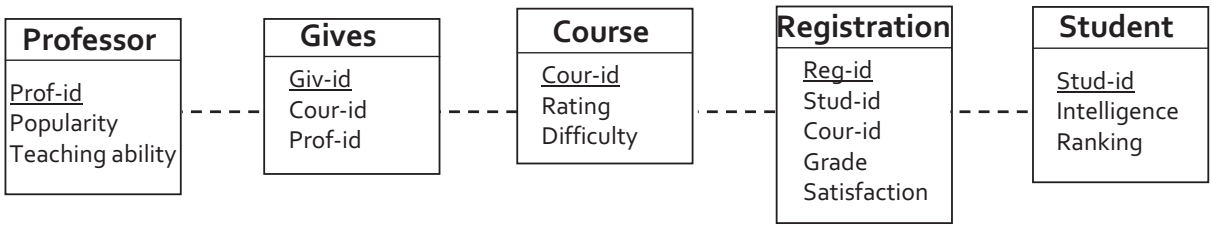


Figure 3.5: The relational schema of the university domain with respect to the DAPER representation

3.2.3.2 Maier's relational model

Maier (Maier et al., 2013c) redefines relational models using the ER model rather than relational schemas. All definitions presented in this section are extracted from (Maier et al., 2013c; Maier, 2014).

Definition 3.2.11. ***Relational schema.*** *A relational schema $\mathcal{S} = (\mathcal{E}, \mathcal{R}, \mathcal{A}, card)$ consists of a set of entity classes $\mathcal{E} = \{E_1, \ldots, E_m\}$; a set of relationship classes $\mathcal{R} = \{R_1, \ldots, R_n\}$, where each $R_i = \langle E_1^i, \ldots, E_{a_i}^i \rangle$, with $E_j^i \in \mathcal{E}$; and a_i is the arity for R_i; a set of attribute classes $\mathcal{A}(I)$ for each item class $I \in \mathcal{E} \cup \mathcal{R}$; and a cardinality function card : $\mathcal{R} \times \mathcal{E} \to \{ONE, MANY\}$.*

Definition 3.2.12. ***Relational skeleton.*** *A relational skeleton σ for a relational schema $\mathcal{S} = (\mathcal{E}, \mathcal{R}, \mathcal{A}, card)$ specifies a set of entity instances $\sigma(E)$ for each $E \in \mathcal{E}$ and relationship instances $\sigma(R)$ for each $R \in \mathcal{R}$. Relationship instances adhere to the cardinality constraints of $\mathcal{S}$: If $card(R, E) = ONE$), then for each $e \in \sigma(E)$ there is at most one $r \in \sigma(R)$ such that e participates in r.*

Definition 3.2.13. ***Relational path.*** *A relational path $[I_j, \ldots, I_k]$ for a relational schema $\mathcal{S}$ is an alternating sequence of entity and relationship classes $I_j, \ldots, I_k \in \mathcal{E} \cup \mathcal{R}$ such that:*

1. *For every pair of consecutive item classes $[E, R]$ or $[R, E]$ in the path, $E \in R$*
2. *For every triple of consecutive item classes $[E, R, E']$, $E \neq E'$.*
3. *For every triple of consecutive item classes $[R, E, R']$, if$R = R'$, then $card(R, E) = MANY$.*

*I_j is called the **base item**, or **perspective**, of the relational path.*

Definition 3.2.14. ***Terminal set.*** *For each skeleton $\sigma \in \sum_{\mathcal{S}}$ and $i_j \in \sigma(I_j)$, the terminal set $P|i_j$ for relational path $P = [I_j, \ldots, I_k]$ of length n is defined inductively as*

$$P^1|i_j = [I_j]\,|i_j = \{i_j\}$$
$$\vdots$$
$$P^n|i_j = [I_j, \ldots, I_k]\,|i_j = \bigcup_{i_m \in P^{n-1}|i_j} \{i_k | ((i_m \in i_k if I_k \in \mathcal{R}) \vee (i_m \in i_k if I_k \in \mathcal{E})) \wedge i_k \notin \bigcup_{l=1}^{n-1} P^l|i_j\}$$

Definition 3.2.15. ***Relational variable.*** *A relational variable $[I_j, \ldots, I_k].X$ consists of a relational path $[I_j, \ldots, I_k]$ and an attribute class $X \in \mathcal{A}(I_k)$.*

Definition 3.2.16. ***Relational variable instance.*** *For each skeleton $\sigma \in \sum_{\mathcal{S}}$ and $i_j \in \sigma(I_j)$, a relational variable instance $[I_j, \ldots, I_k].X|i_j$ for a relational variable $[I_j, \ldots, I_k].X$ is the set of random variables $\{i_k.X | X \in \mathcal{A}(I_k) \wedge i_k \in [I_j, \ldots, I_k]\,|i_j \wedge i_k \in \sigma(I_k)\}$.*

Definition 3.2.17. ***Relational dependency.*** *A relational dependency $[I_j, \ldots, I_k].Y \to [I_j].X$ is a directed probabilistic dependence from attribute class Y to X through the relational path $[I_j, \ldots, I_k]$.*

Definition 3.2.18. ***Relational model.*** *A relational model $\mathcal{M}_\Theta$ consists of two parts:*

1. *The structure $\mathcal{M} = (\mathcal{S}, \mathcal{D})$: a schema $\mathcal{S}$ paired with a set of relational dependencies $\mathcal{D}$ defined over $\mathcal{S}$.*

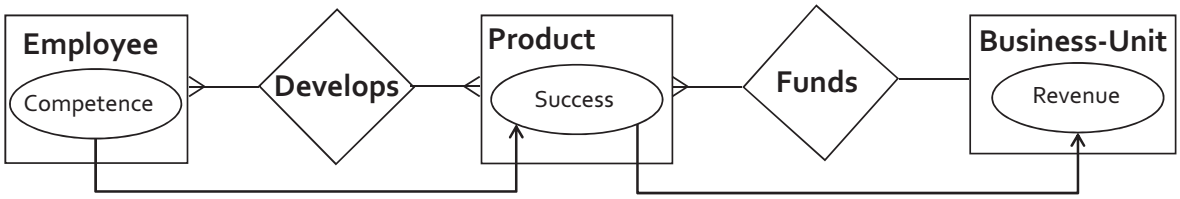


Figure 3.6: Example of a relational model for the organization domain (Maier et al., 2013b)

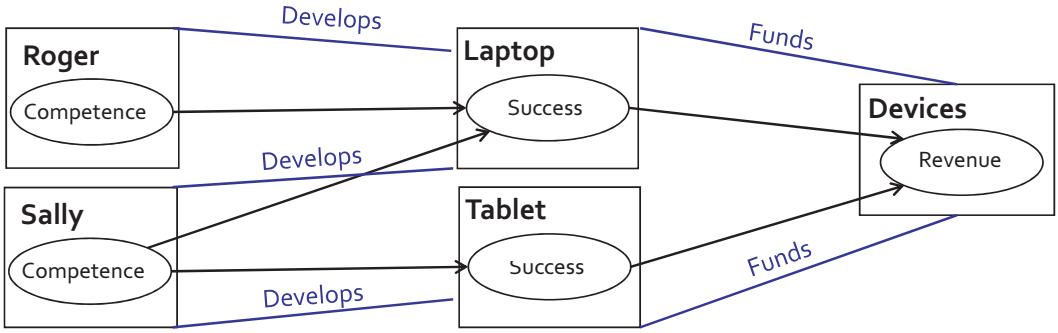


Figure 3.7: Example of a ground graph for the organization domain (Maier et al., 2013b)

2. *The parameters Θ: a conditional probability distribution $P([I_j].X|parents([I_j].X))$ for each relational variable of the form $[I_j].X$, where $I_j \in \mathcal{E} \cup \mathcal{R}$, $X \in \mathcal{A}(I_j)$ and $parents([I_j].X) = \{[I_j,\ldots,I_k].Y|\,[I_j,\ldots,I_k].Y \rightarrow [I_j].X \in \mathcal{D}\}$ is the set of parent relational variables.*

Definition 3.2.19. ***Ground graph.*** *The ground graph $\mathcal{GG}_{\mathcal{M}\sigma} = (V,E)$ for a relational model structure $\mathcal{M} = (\mathcal{S},\mathcal{D})$ and a skeleton $\sigma \in \sum_{\mathcal{S}}$ is a directed graph with nodes $V = \{i.X|I \in \mathcal{E} \cup \mathcal{R} \wedge i \in \sigma(I)\}$ and edges $E = \{i_k.Y \rightarrow i_j.X|i_k.Y, i_j.X \in V \wedge i_k.Y \in [I_j,\ldots,I_k].Y|i_j \wedge [I_j,\ldots,I_k].Y \rightarrow [I_j].X \in \mathcal{D}\}$.*

Definition 3.2.20. ***Class dependency graph.*** *The class dependency graph $\mathcal{G}_{\mathcal{M}} = (V,E)$ for a relational model structure $\mathcal{M} = (\mathcal{S},\mathcal{D})$ is a directed graph with a node for each attribute of every item class $V = \{I.X|I \in \mathcal{E} \cup \mathcal{R} \wedge X \in \mathcal{A}(I)\}$ and edges between pairs of attributes supported by relational dependencies in the model $E = \{I_k.Y \rightarrow I_j.X|\,[I_j,\ldots,I_k].Y \rightarrow [I_j].X \in \mathcal{D}\}$.*

Example 3.2.4. *Figure 3.6 is an example of a relational model for the organization domain. The example contains 3 entity classes $\mathcal{E} = \{Employee, Product, Business-Unit\}$ and 2 relationship classes $\mathcal{R} = \{Develops, Funds\}$. Employees cause the success of products they develop. The success of products influences the revenue received by the business unit funding the product. The dependencies are specified by relational paths as follows: $[Product, Develops, Employee].Competence \rightarrow [Product].Success$ and $[Business-Unit, Funds, Product].Success \rightarrow [Business-Unit].Revenue$.*

Figure 3.7 shows the ground graph for the relational model in Figure 3.6 applied to a relational skeleton consisting of two employees, two products, and a single business unit.

3.2.4 About relational d-separation

Maier et al. (Maier et al., 2013c) have developed the relational extension of the Markov condition:

Definition 3.2.21. ***Relational Markov condition.*** *Let X be a relational variable for perspective $B \in \mathcal{E} \cup \mathcal{R}$ defined over relational schema $\mathcal{S}$. Let $Nd(X)$ be he non-descendant variables of X, and let $Pa(X)$ be the set of parent variables of X. Then, for relational model $\mathcal{M}_\Theta$, $P(X|Nd(X),Pa(X)) = P(X|Pa(X))$ if and only if $\forall x \in X|bP(x|Nd(x),Pa(x)) = P(x|Pa(x))$ in parameterized ground graph $\mathcal{GG}_{\mathcal{M}_\Theta}$ for all skeleton $\sigma \in \sum_{\mathcal{S}}$ and for all $b \in \sigma(B)$.*

In other words, a relational variable X is independent of its non-descendants given its parents if and only if, for all possible parameterized ground graphs, the Markov condition holds for all instances of X (Maier et al., 2013c). The authors have also proved that the d-separation criterion cannot be reproduced as it is (cf. Section 1.3.1) in the relational context.

In contrast to standard Bayesian networks, a relational model, such as a RBN, is a meta-model, a template that results in several ground graphs (each of which is a BN) with different structures that depend on the used data instantiation. So, a similar theory that allows to reason about relational d-separation has to be provided. Maier et al. (Maier et al., 2013c) have developed the relational d-separation theory through a new representation that they called *abstract ground graph (AGG)* and they proved how this latter enables a sound, complete, and computationally efficient method for answering d-separation queries about relational models (Maier et al., 2013c).

An abstract ground graph abstracts over all possible ground graphs, using only knowledge about the schema and the model (entities, cardinalities and dependencies), with a specified maximum slot chain length h [1]. An abstract ground graph is invariant of the size of ground graphs even though these latter can be arbitrarily large. An AGG has as nodes as nodes all possible relational variables that can be reached from a perspective class, with a slot chain length less than or equal to h, as well as nodes representing intersection between pairs of variables that may intersect. AGG edges are between pairs of relational variables and edges inherited from both relational variables sources of every intersection variables.

Definition 3.2.22. ***Abstract ground graph.*** *An abstract ground graph $\mathcal{AGG}_{\mathcal{M}B} = (V, E)$ for a relational model structure $\mathcal{M} = (\mathcal{S}, \mathcal{D})$ and a perspective $B \in \mathcal{E} \cup \mathcal{R}$ is a directed graph that abstracts the dependencies $\mathcal{D}$ for all ground graphs $\mathcal{GG}_{\mathcal{M}\sigma}$, where $\sigma \in \sum_{\sigma}$.*
The set of nodes in $\mathcal{AGG}_{\mathcal{M}B}$ is $V = RV \cup IV$, where

- *RV is the set of all relational variables of the form $[B, \ldots, I_j].X$*
- *IV is the set of all pairs of relational variables that could have non-empty intersections (referred to as intersection variables)*
  $IV = \{RV_1 \cap RV_2 | RV_1, RV_2 \in RV \wedge RV_1$
  $= [B, \ldots, I_k, \ldots, I_j].X \wedge RV_2$
  $= [B, \ldots, I_l, \ldots, I_j].X \wedge I_k \neq I_l\}$

The set of edges in $\mathcal{AGG}_{\mathcal{M}B}$ is $E = RVE \cup IVE$, where

- *$RVE \subset RV \times RV$ is the set of edges between pairs of relational variables such that:*
  $RVE = \{[B, \ldots, I_k].Y \rightarrow [B, \ldots, I_j].X | [I_j, \ldots, I_k].Y \rightarrow [I_j].X \in \mathcal{D} \wedge [B, \ldots, I_k]$
  $\in extend([B, \ldots, I_j], [I_j, \ldots, I_k])\}$
- *$IVE \subset IV \times RV \cup RV \times IV$ is the set of edges inherited from both relational variables involved in every intersection variable in IV:*
  $IVE = \{\hat{r} \rightarrow [B, \ldots, I_j].X | \hat{r} = P_1.Y \cap P_2.Y \in IV \wedge (P_1.Y \rightarrow [B, \ldots, I_j].X \in RVE$
  $\vee P_2.Y \rightarrow [B, \ldots, I_j].X \in RVE)\}$
  $\cup$
  $\{[B, \ldots, I_k].Y \rightarrow \hat{X} | \hat{X} = P_1.X \cap P_2.X \in IV \wedge ([B, \ldots, I_k].Y \rightarrow P_1.X \in RVE$
  $\vee [B, \ldots, I_k].Y \rightarrow P_2.X \in RVE)\}$

The *extend* method (cf. Definition 3.2.23) allows to construct the set of all valid relational paths from two input relational paths. It is called repeatedly during the creation of an abstract ground graph as it allows to capture all paths of dependence among the random variables in the relational variable instances for all represented ground graphs (Maier et al., 2013c).

Definition 3.2.23. ***Extending relational paths.*** *Let P_{orig} and P_{ext} be two relational paths for schema $\mathcal{S}$. Let n_o be the length of P_{orig} and n_e be the length of P_{ext}. $P^{i,j}$ corresponds to 1-based i-inclusive, j-inclusive*

1. Following Maier et al. (Maier et al., 2013c), h is called hop threshold

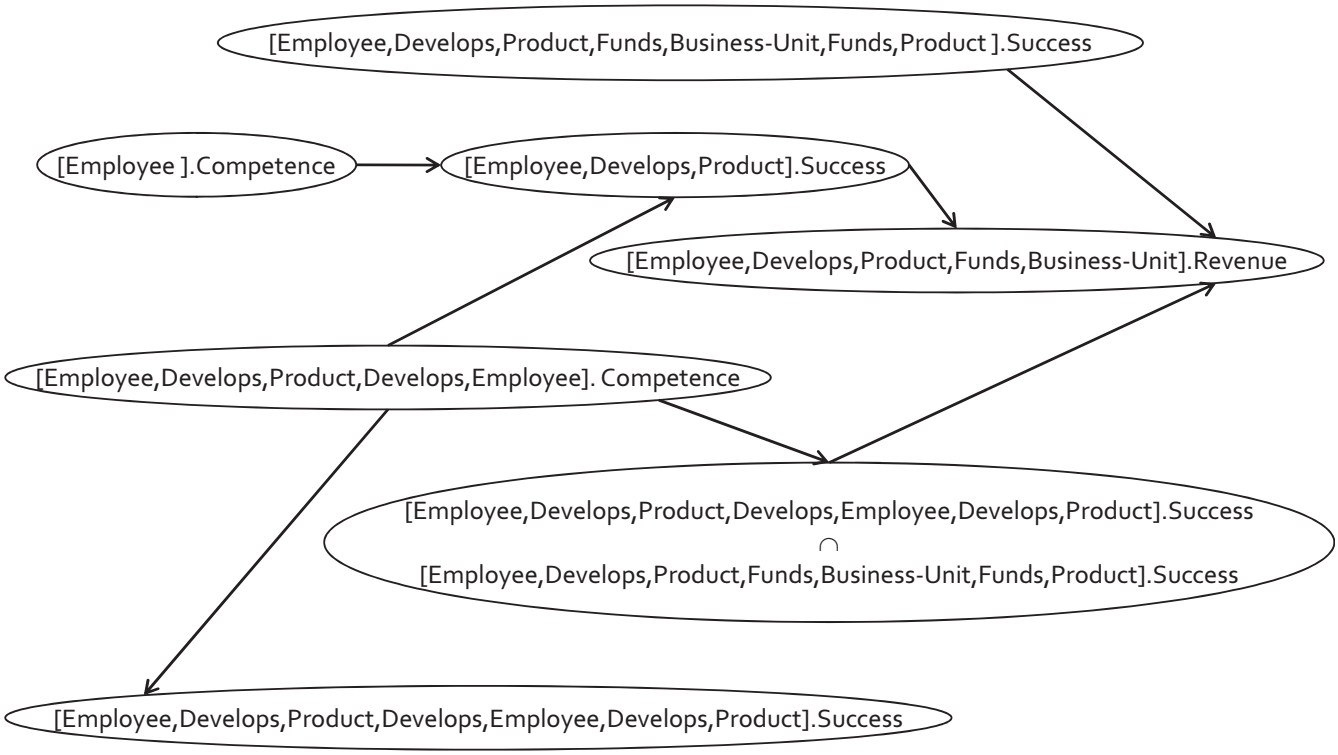


Figure 3.8: An abstract ground graph for the organization domain model in Figure 3.6 from the Employee perspective and with a hop threshold =6 (Maier et al., 2013b)

sub-path indexing, + is concatenation of paths, and reverse is a method that reverses the order of the path. To extend P_{orig} with P_{ext}, the following three functions are needed:

i. $pivot(P_1, P_2) = \{i | P_1^{1,i} = P_2^{i,1}\}$

ii. $\textit{is valid}(P) = \begin{cases} \textit{True if P does not violate Definition 3.2.13} \\ \textit{False otherwise} \end{cases}$

iii. $extend(P_{orig}, P_{ext}) = \{P_{orig}^{1,n_o-i+1} + P_{ext}^{i+1,n_e} | i \in pivots(reverse(P_{orig}), P_{ext}) \wedge \textit{is valid}(P)\}$

Example 3.2.5. *Figure 3.8 shows the abstract ground graph for the organization domain modeled by Figure 3.6, from the Employee perspective with hop threshold = 6. It includes 6 relational variables and one intersection node:*

- $RV = \{[Employee].Competence, [Employee, Develops, Product, Funds, Business{-}Unit, Funds, Product].Success, [Employee, Develops, Product].Success, [Employee, Develops, Product, Funds, Business{-}Unit].Revenue, [Employee, Develops, Product, Develops, Employee].Competence, [Employee, Develops, Product, Develops, Employee, Develops, Product].Success\}$
- $IV = \{([Employee, Develops, Product, Develops, Employee, Develops, Product].Success \cap [Employee, Develops, Product, Funds, Business - Unit, Funds, Product].Success)\}$

It contains 7 edges, with:

- $RVE = \{[Employee].Competence \rightarrow [Employee, Develops, Product].Success, [Employee, Develops, Product, Funds, Business{-}Unit, Funds, Product].Success) \rightarrow [Employee, Develops, Product, Funds, Business{-}Unit].Revenue, [Employee, Develops, Product].Success \rightarrow [Employee, Develops, Product, Funds, Business{-}Unit].Revenue, [Employee, Develops, Product, Develops, Employee].Competence \rightarrow [Employee, Develops, Product].Success, [Employee, Develops, Product,$

$Develops, Employee].Competence \rightarrow [Employee, Develops, Product, Develops, Employee, Develops, Product].Success\}$

- $IVE = \{([Employee, Develops, Product, Develops, Employee, Develops, Product].Success \cap [Employee, Develops, Product, Funds, Business{-}Unit, Funds, Product].Success) \rightarrow [Employee, Develops, Product, Funds, Business{-}Unit].Revenue, [Employee, Develops, Product, Develops, Employee].Competence \rightarrow ([Employee, Develops, Product, Develops, Employee, Develops, Product].Success \cap [Employee, Develops, Product, Funds, Business-Unit, Funds, Product].Success)\}$

The extend method has been called several times during the construction of AGG of Figure 3.8. It has been used to insert edges corresponding to direct causes. For instance, $extend(P_{orig}, P_{ext})$*, with* $P_{orig} = [Employee, Develops, Product]$ *and* $P_{ext} = [Product, Develops, Employee]$ *has as* $pivots(reverse(P_{orig}), P_{ext}) = \{1, 2, 3\}$*. The pivot at* 1 *and* 3 *gives as result* $\{[Employee], [Employee, Develops, Product, Develops, Employee]\}$*, which leads to the insertion of two edges in the AGG:* $[Employee].Competence \rightarrow [Employee, Develops, Product].Success$ *and* $[Employee, Develops, Product, Develops, Employee].Competence \rightarrow [Employee, Develops, Product].Success$*. However the pivot at* 2 *gives the invalid relational path* $[Employee, Develops, Employee]$.

Abstract ground graphs are used to reason about relational d-separation queries. As the construction of an AGG depends on the chosen perspective, then, for a given relational model, there is as many AGG as classes. Each AGG defined with respect to a given perspective can then be used to reason about relational d-separation queries for that perspective. The relational d-separation is defined as follows (Maier et al., 2013c):

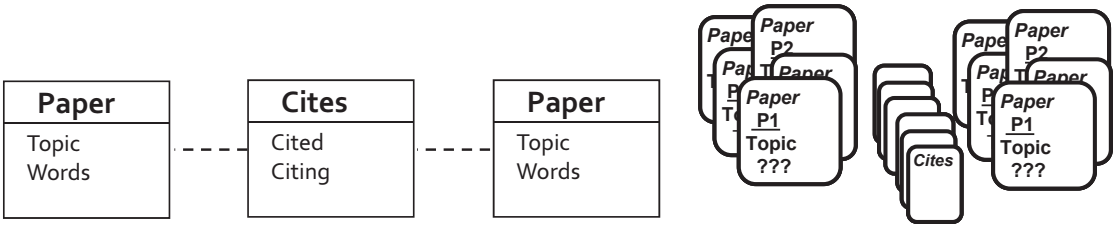
Definition 3.2.24. ***Relational d-separation.*** *Let* X, Y*, and* Z *be three distinct sets of relational variables with the same perspective* $B \in \mathcal{X}$ *defined over relational schema* $\mathcal{R}$*. Then, for relational model structure* $\mathcal{S}$*,* X *and* Y *are d-separated by* Z *if and only if, for all skeletons* $\sigma \in \sum_{\mathcal{R}}$*,* $X|b$ *and* $Y|b$ *are d-separated by* $Z|b$ *in ground graph* $GG_{\mathcal{S}\sigma}$ *for all* $b \in \sigma(B)$*.*

Example 3.2.6. *For relational models, we need to instantiate the model to be able to perform probabilistic inference. So d-separation can be applied at the ground graph level.*

- *d-separation applied directly to the model in Figure 3.6 states that employee competence is conditionally independent of the revenue of business units given the success of products.*
- *d-separation applied to the ground graph of Figure 3.7 states that to d-separate employee competence from business unit revenue, we should condition on both the success of developed products and the competence of other employees who work on those products.*
- *d-separation applied to the abstract ground graph of Figure 3.8, states that the conditioning set must include both product success (*$[Employee, Develops, Product].Success$*) and the competence of other employees developing the same products (*$[Employee, Develops, Product, Develops, Employee].Competence$*).*

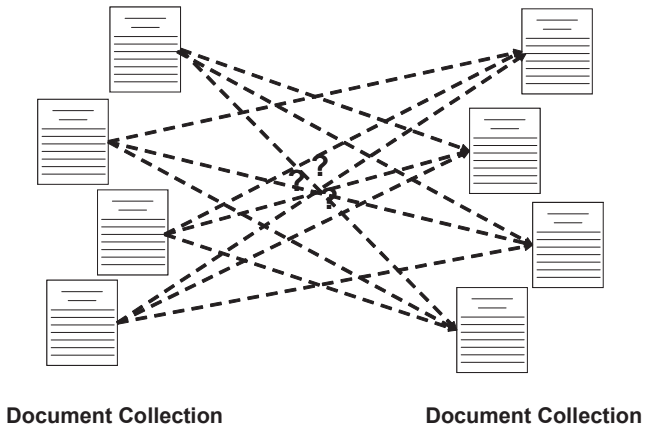
3.2.5 Reasoning with relational Bayesian networks

Inference in RBNs and similar models is performed at the ground graph level. By this way, standard inference algorithms for Bayesian networks can be used to query the GBN (cf. Section 1.3.4). However, GBNs can be very large and complex, which makes the inference process very expensive. On the other hand, inferring this way leads to the loss of the RBN structure, as it is converted into a flat BN representation. (Pfeffer, 2000; Pfeffer and Koller, 2000) have discussed these issues and have proposed to perform inference in the lifted (i.e. non-grounded) model. Several inference algorithms have been proposed in this direction (Pfeffer and Koller, 2000; Milch et al., 2008; Kisynski and Poole, 2009; Gonzales and Wuillemin, 2011; Wuillemini and Torti, 2012).



(a) Relational schema for the citation domain

(b) Reference uncertainty: An object skeleton for the citation domain



(c) Existence uncertainty in the citation domain

Figure 3.9: Examples of structured uncertainty (Getoor et al., 2007)

3.2.6 Structural uncertainty

One of the highlights of RBNs, which differentiates them from classical BNs, is its ability to represent structural uncertainty. In fact, BNs allow to describe a world using a set of descriptive attributes and assign conditional probability distributions to them to represent uncertainty. However, BNs do not encode uncertainty about the existence of probabilistic relations. In RBNs, as defined till now, only the descriptive attributes are uncertain. All relations between attributes are given by the relational skeleton and all values of reference slots are specified. This assumption implies that the model cannot be used to predict the relational structure itself. Yet, the relational structure is informative in and of itself (Getoor et al., 2007). Pfeffer provides two extensions of RBNs to model more complex structural uncertainty namely, reference uncertainty and existence uncertainty (Pfeffer, 2000), also known as probabilistic models of link structure.

- **RBN with reference uncertainty.** This representation assumes that a partial skeleton called *object skeleton* is given instead of a full one. This partial skeleton specifies only the objects in each class X but not the values of the reference slots.
- **RBN with existence uncertainty.** In this representation no assumptions about the number of links that exist are made. The number of links that exist and the identity of the links are all part of the probabilistic model and can be used to make inferences about other attributes in the model.

Example 3.2.7. *let us consider example of Figure 3.9, reference uncertainty model assumes that all objects are specified and search for relations among them, i.e., citation links between papers (see Figure 3.9(b)).*
Figure 3.9(c) illustrates existence uncertainty, the goal of such a representation is to provide an explicit model for the presence or absence of citations. Each potential citation can be present or absent. Unlike the reference uncertainty model, the number of citation is not known in advance.

3.3 RBN and similar models structure learning

RBNs construction states finding the dependency structure $\mathcal{S}$ and the parameters for a given relational schema $\mathcal{R}$. Given the dependency structure $\mathcal{S}$, parameters learning consists on providing $\theta_{\mathcal{S}}$. The key concept in parameter estimation is the likelihood function known to be the probability of the data given the model:

$$P(\theta_{\mathcal{S}}|\mathcal{I}, \sigma, \mathcal{S}) = P(\mathcal{I}|\mathcal{S}, \sigma, \theta_{\mathcal{S}}) \tag{3.2}$$

It is the likelihood function of standard BNs. The main difference lies on the fact that there may be nodes having the same probability distributions induced by the RBN given the relational skeleton. RBN parameter estimation is then performed either using the statistical approach, namely maximum likelihood parameter estimation, or using the Bayesian approach. The sufficient statistics are the counts of the number of times that we observe the child states given its parents states. These counts have to be performed on a complete instance $\mathcal{I}$, using SQL queries.

The dependency structure identification is inspired from classical methods of finding standard BNs structure from propositional data. Only few works have been presented in this area. These latter concern either RBNs or other similar representations (cf. Section 3.2.3) structure learning. As these models are closely related, this section is dedicated to present the relational state-of-the-art extensions of the score-based and constraint-based techniques for either RBNs or other similar model representations. Then, we will discuss hybrid relational approaches.

3.3.1 RBN structure learning

Friedman et al. (Friedman et al., 1999a) have proposed a relational extension of a score-based approach. To the best of our knowledge, this algorithm is the only proposed work in this area. Similar to BNs Score-based approaches, the relational score-based algorithm assigns a score to each possible network, to search after through the space of candidate graphs for the network that maximizes this score. Which structures are legal? How to evaluate them? How to proceed in order to find the high-scoring structure? These three main issues are discussed by Getoor in (Getoor, 2002).

3.3.1.1 Finding legal structures

The dependency structure $\mathcal{S}$ has to be acyclic for any possible skeleton σ_r. Thus, for a RBN Π, a class dependency $\mathcal{G}_\Pi$ is defined. This graph considers potential dependencies at the class level. We have to maintain models whose dependency structures are stratified. $\mathcal{S}$ is stratified if the colored class dependency graph is stratified, that is, if every cycle in the graph contains at least one green edge and no red edges. Specifying that certain reference slots are guaranteed acyclic represents a prior knowledge allowing us to guarantee the legality of certain dependency models.

3.3.1.2 Scoring function

As for standard Bayesian networks, a score-based approach is feasible in practice, if the used score is locally decomposable, i.e., it can be expressed as the sum of local scores at each node.

For RBNs the posterior probability of the structure $\mathcal{S}$ given an instantiation $\mathcal{I}$ is :

$$P(\mathcal{S}|\mathcal{I}, \sigma) \propto P(\mathcal{S}|\sigma)P(\mathcal{I}|\mathcal{S}, \sigma) \tag{3.3}$$

This score is composed of two parts:

– The prior probability of the structure $P(\mathcal{S}|\sigma)$, which is equal to $P(\mathcal{S})$ as the choice of structure is independent of the skeleton. This prior will penalize long indirect slot chains denoted by $length_{\mathcal{K}}$:

$$logP(\mathcal{S}) = \sum_{i} \sum_{A \in \mathcal{A}(X_i)} \sum_{u \in V(Pa(X_i.A))} length_{\mathcal{K}}(X_i.A, Pa(X_i.A)) \tag{3.4}$$

Algorithm 14 Relational Greedy search (Friedman et al., 1999a)

Require: $\mathcal{R}$: A relational schema, $\mathcal{S}$: A prior dependency structure, $\mathcal{I}$: A database instance, K_{max}: the maximum slot chain length.
Ensure: The local optimal dependency graph $\mathcal{S}$.

repeat
  $Max_{score} \leftarrow Score(\mathcal{S})$
  repeat
    $List\_neighbors \leftarrow Generate\_neighborhood(Current\_SlotChain\_length)$
    $\mathcal{S}_{new} \leftarrow Argmax_{\mathcal{S}' \in neighborhood_{\mathcal{S}}}(Score(\mathcal{S}'))$
    if $Score(\mathcal{S}_{new}) \geq Max_{score}$ **then**
      $Max_{score} \leftarrow score(\mathcal{S}_{new})$
      $\mathcal{S} \leftarrow \mathcal{S}_{new}$
    end if
  until No change
  $Current\_SlotChain\_length \leftarrow Current\_SlotChain\_length + 1$
until No changes or $Current\_SlotChain\_length = K_{max}$

– The probability of the data assuming that structure $P(\mathcal{I}|\mathcal{S}, \sigma)$, which is equal to the marginal likelihood of $\mathcal{I}$ given $\mathcal{S}$:

$$P(\mathcal{I}|\mathcal{S}, \sigma) = \prod_i \prod_{A \in \mathcal{A}(X_i)} \prod_{u \in V(Pa(X_i.A))} DM(\{C_{X_i.A}[v, u]\}, \{\alpha_{X_i.A}[v, u]\}) \tag{3.5}$$

Where $DM(\{C_{X_i.A}[v, u]\}, \{\alpha_{X_i.A}[v, u]\}) = \frac{\Gamma(\sum_v \alpha[v])}{\Gamma(\sum_v (\alpha[v]+C[v]))} \prod_v \frac{\Gamma(\alpha[v]+C[v])}{\Gamma(\alpha[v])}$,
and $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$ is the Gamma function.
As usual, we use the log of this function

$$logP(\mathcal{I}|\mathcal{S}, \sigma) = \sum_i \sum_{A \in \mathcal{A}(X_i)} \sum_{u \in V(Pa(X_i.A))} log[DM(\{C_{X_i.A}[v, u]\}, \{\alpha_{X_i.A}[v, u]\})] \tag{3.6}$$

Consequently, the Relational BD score is as follows:

$$\mathcal{R}BD_{score} = \sum_i \sum_{A \in \mathcal{A}(X_i)} \sum_{u \in V(Pa(X_i.A))} log[DM(\{C_{X_i.A}[v, u]\}, \{\alpha_{X_i.A}[v, u]\})] - \sum_i \sum_{A \in \mathcal{A}(X_i)} \sum_{u \in V(Pa(X_i.A))} length_{\mathcal{K}}(X_i.A, Pa(X_i.A)) \tag{3.7}$$

Friedman et al. (Friedman et al., 1999a) have used a relational extension of the BD score instead of a relational extension of the $BDeu$ score since unlike BNs, the notion of Markov equivalence class is not yet developed for RBNs.

3.3.1.3 Search procedure.

The search procedure proposed in (Friedman et al., 1999a) resorts to heuristic search, namely the greedy hill-climbing search. In order to reduce both the number of possible structures, which is usually very large or even infinite, and database querying, which is quite expensive, the greedy algorithm proceeds in phases.

The choice of $Pot_k(X.A)$ is based on classes dependencies: At phase 0 only dependencies within a class are considered, At Phase 1, only dependencies from classes related to X by a reference slot are considered. At phase 2 dependencies from classes related to X using slot chain of length equal 2 are considered, etc. The algorithm stops when no changes are detected or when the maximum slot chain length is reached.

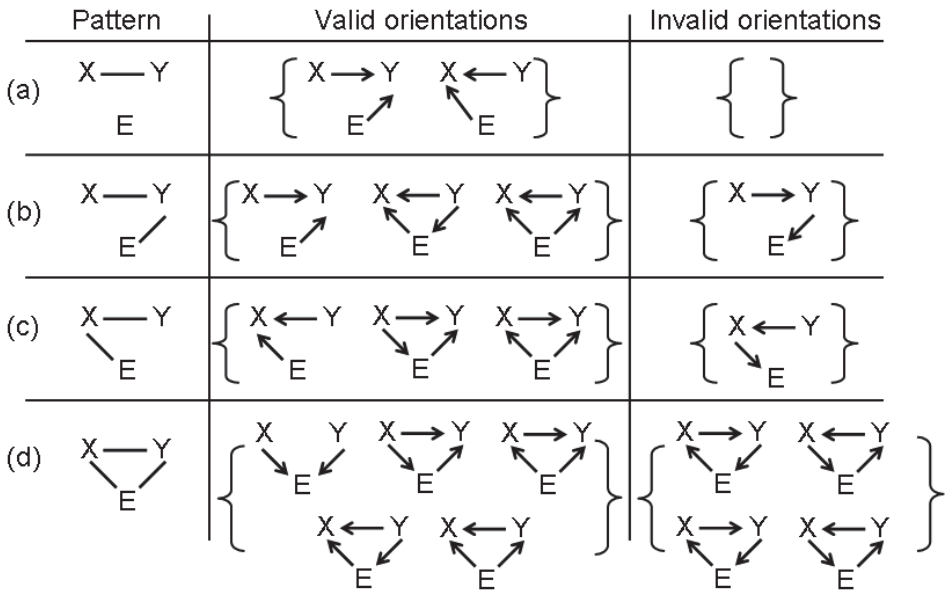


Figure 3.10: RPC algorithm: the four new constraints, where E is an existence attribute and X and Y are two unit attribute classes (Maier et al., 2010).

The neighbor generation process consists in applying the add_edge, $delete\_edge$ and $reverse\_edge$ operators between a target variable $X.A$ and a variable $Y \in Pot_K(X.A)$ where $Pot_K(X.A)$ is the set of potential parents of $X.A$ given a slot chain length. $Pot_K(X.A)$ may contain aggregate variables depending on the crossed slot chain. Keeping one operator goes through a verification process in order to maintain the class dependency graph acyclic. It is clear that deleting an edge does not present a concern, so this verification concerns only adding and reversing edges. The verification is performed using the $Depth\_First\_Search$ algorithm for graph traversing. The relational greedy search algorithm is depicted in algorithm 14.

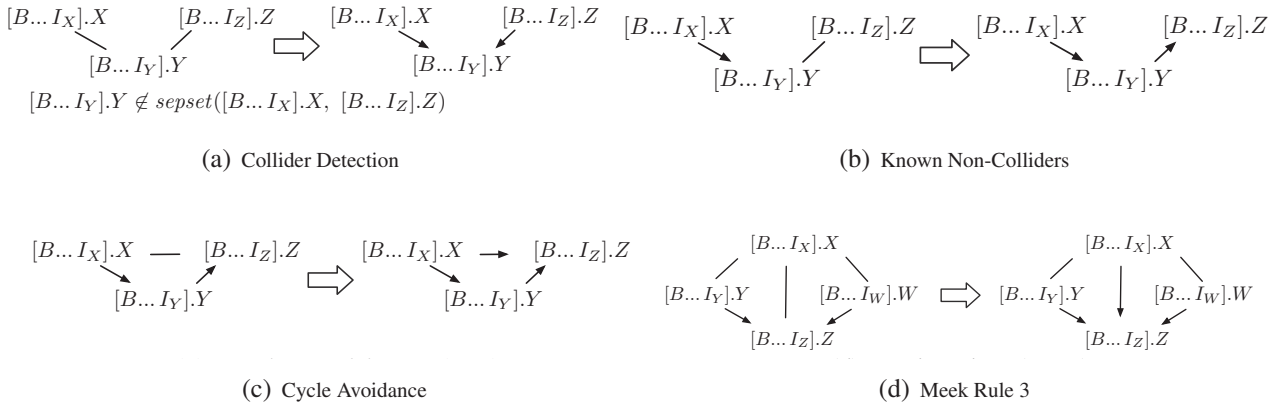
3.3.2 Similar models structure learning

In (Maier et al., 2010), the authors proposed a relational extension of the PC algorithm (cf. Section 1.4.2). The authors proposed a constraint-based approach to learn DAPER structure from relational data. They preserve the same rules for the RPC algorithm, to which they add four new constraints specific to the relational learning process, which are depicted in Figure 3.11.

First, they identify the key differences of relational data with propositional data:

- Variable space: RPC takes as input a ER model represented in first-order logic and a DB instance which is an instantiation of this ER and returns a partially directed DAPER. To perform relational learning, some modifications have to be done: RPC uses new concepts, which are:
  - The set of unit classes $\mathcal{U} = \{U_1, \ldots, U_k\}$: $\mathcal{U} \subset (\mathcal{E} \cup \mathcal{R})^+$, where each U consists of one or more entity or relationship classes which are relationally connected. U is defined over a base entity or relationship class, $b(U)$.
  - The unit attribute class: is any attribute class defined over an entity or relationship class within the unit. $\mathcal{A}(U) = \bigcup_{B \in U} \mathcal{A}(B)$.
  - Potential causal dependency: is composed of a treatment variable $\mathcal{T} \in \mathcal{A}(U)$ and an outcome variable $\mathcal{O} \in \mathcal{A}(b(U))$.

  RPC requires identifying the set of possible common causes for treatment and outcome variables of a potential dependency $\langle A.X, B.Y \rangle$, which is the union of the set of treatment variables when $A.X$ and $B.Y$ are considered outcomes $\{C.Z | \langle C.Z, A.X \rangle\} \cup \{D.Z | \langle D.Z, B.Y \rangle\}$.
- Aggregates and asymmetry: RPC uses aggregation functions to create a single value for each unit attribute $f(A.X)$. RPC tests the association from both perspective $\langle f(X), Y \rangle$ and $\langle f(Y), X \rangle$ (as



(a) Collider Detection (b) Known Non-Colliders

(c) Cycle Avoidance (d) Meek Rule 3

Figure 3.11: RCD orientation rules on an abstract ground graph from perspective B (Maier et al., 2013a)

there is an inherent asymmetry for pairs of unit attributes because of the aggregation requirement), if a statistical association is detected in either, then an association exists between X and Y.

– Structural variables: RPC includes an existence attribute for each relationship class. If this attribute is viewed as a treatment variable, then RPC uses the count aggregate function to represent cardinality. If it is viewed as an outcome variable, RPC tests for association between a treatment variable and the existence of the relationship.

Unlike the PC algorithm which is sound and complete the RPC algorithm did not satisfy these criteria. In fact, in (Maier et al., 2013c), the authors have discussed that in order to adapt the constraint based as well as the hybrid approaches of BNs structure learning to learn from relational data, a similar theory that allows to reason about relational d-separation has to be provided. They invented the AGG notion (cf. Section 3.2.4) and used it in (Maier et al., 2013a) to propose a new constraint-based algorithm, called relational causal discovery (RCD) algorithm.

This algorithm makes the underlying assumptions:

– Causal sufficiency.
– Faithfulness.
– Sufficient maximum hop threshold.
– Sufficient depth.
– Perfect conditional independence tests.

Definition 3.3.1. ***Bivariate orientation rule.*** *Let Π be a relational model and G a partially directed abstract ground graph for Π, perspective $X_i \in \mathcal{X}$, and maximum slot chain length h. If $X_i.A_{\in\mathcal{A}(X_i)}[X_i, \ldots, X_j].A_{\in\mathcal{A}(X_j)}$ is in G, and $X_i.A \perp [X_i, \ldots, X_j, \ldots, X_i].A | Z$, then* [2]

– *if $[X_i, \ldots, X_j].A_{\in\mathcal{A}(X_j)} \in Z$, orient as: $X_i.A \leftarrow [X_i, \ldots, X_j].A_{\in\mathcal{A}(X_j)}$*
– *if $[X_i, \ldots, X_j].A_{\in\mathcal{A}(X_j)} \notin Z$, orient as: $X_i.A \rightarrow [X_i, \ldots, X_j].A_{\in\mathcal{A}(X_j)}$*

The RCD algorithm, outlined by Algorithm 15 performs in two phases (Algorithm 15). In the first phase, given a hop threshold, RCD starts by providing the set of all potential dependencies, then continues by removing conditional independences found using conditional independence tests. RCD uses the linear regression as statistical association measurement. When performing statistical tests, RCD verifies whether a statistical association is detected between two variables in both directions and it leaves the dependence if a statistical association exists in at least one direction. This assumption is nearly similar to the *OR* condition of the MBOR algorithm for BNs (cf. Section 1.4.4). It allows to provide a broader list of dependencies between variables.

In the second phase, RCD determines the orientation of the remaining dependencies. It preserves the same orientation rules as the PC algorithm at the level of abstract ground graphs and add only one additional

2. This rule has to be performed if $card([X_i, \ldots, X_j]) = MANY$.

Algorithm 15 Relational Causal Discovery(RCD) (Maier et al., 2013a)

Require: $\mathcal{R}$: A relational schema, h: A hopThreshold, $\mathcal{I}$: A database instance, depth.
Ensure: A set of canonical dependencies.

$PDs \leftarrow getPotentialDeps(\mathcal{R}, h)$
$N \leftarrow initializeNeighbors(\mathcal{R}, h)$
$S \leftarrow \{\}$

Phase 1

for $d \leftarrow 0$ to depth **do**
  for $X \rightarrow Y \in PDs$ **do**
    for each $condSet \in powerSet(N[Y] \backslash \{X\})$ **do**
      if |condSet|=d **then**
        if $X \perp Y | condSet$ **then**
          $PDs \leftarrow PDs \backslash \{X \rightarrow Y, Y \rightarrow X\}$
          $S[X, Y] \leftarrow condSet$
          break
        end if
      end if
    end for
  end for
end for

Phase 2

$AGGs \leftarrow buildAbstractGroundGraph(PDs)$
$AGGs, S \leftarrow ColliderDetection(AGGs, S)$
$AGGs, S \leftarrow BivariateOrientation(AGGs, S)$
repeat
  $AGGs \leftarrow KnownNonColliders(AGGs, S)$
  $AGGs \leftarrow CycleAvoidance(AGGs, S)$
  $AGGs \leftarrow MeekRule3(AGGs, S)$
until No changes
$getCanonicalDependencies(AGGs)$

rule called *bivariate orientation*. Phase 2 starts by constructing as many AGGs as the number of classes following Definition 3.2.22 and using the list of dependencies PDs derived from the first phase. Then, for each AGG, it applies the collider detection rule (cf. Figure 3.11(a)) and the bivariate orientation rule (cf. Definition 3.3.1). This latter leverages relational dependencies that cross relationships with Many-to-Many cardinalities. It allows to detect relational autocorrelation (Jensen and Neville, 2002) and verifies whether a distinct variable is a member of the separating set that eliminates the autocorrelation. Finally phase 2 iterates on the known non-collider (cf. Figure 3.11(b)), cycle avoidance (cf. Figure 3.11(c)) and Meek rule 3 (cf. Figure 3.11(d)) until no improvement is detected.

Even if RCD algorithm aims to meet the PC algorithm (cf. Section 1.4.2) in a relational context, it is unable to return a final graph, equivalent to the PDAG returned by PC, at a relational level. Rather, it constructs a set of AGGs, where each of them is equivalent to a PDAG from a given perspective. Then, it merges all found dependencies into a set of canonical dependencies (cf. Definition 3.3.2). As the set of canonical dependencies is derived from a set of PDAGs, it may contain unoriented dependencies. RCD is sound and complete for causally sufficient relational data. Maier et al. (Maier et al., 2013a) proved that the set of canonical dependencies presents a correct maximally oriented model $\mathcal{M}$ via Theorem 3.3.1.

Definition 3.3.2. ***Canonical dependency.*** *For a canonically specified dependency:*

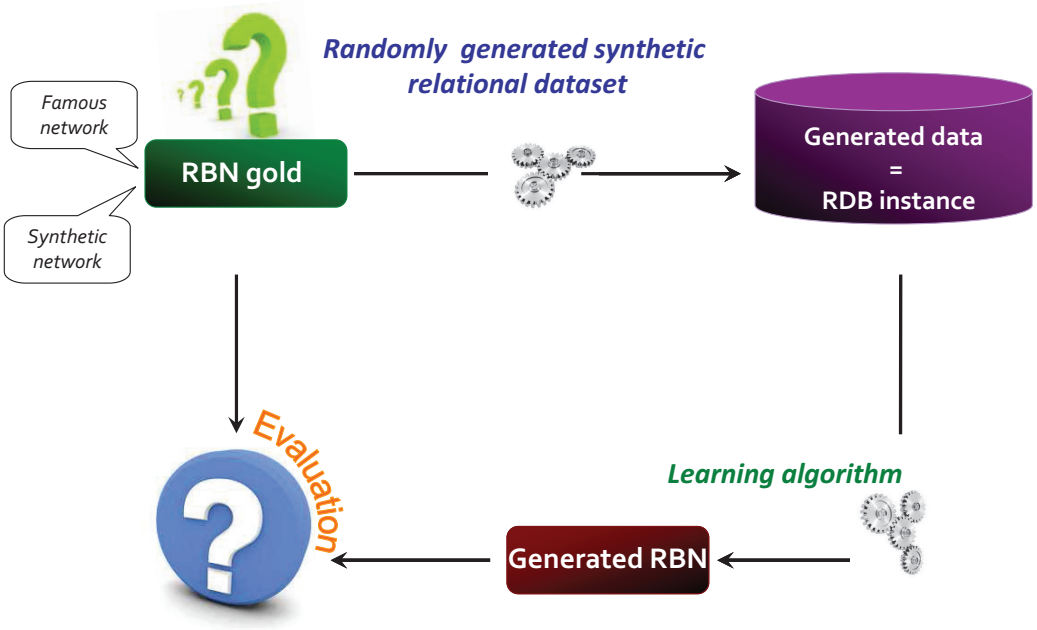


Figure 3.12: Evaluation process of a RBN structure learning algorithm

– *The relational path of the child consists of a single item class: the class to which the child belongs.*
– *The relational path of the parent starts from the class to which the child belongs.*

Theorem 3.3.1. *Given a schema and probability distribution $\mathcal{P}$, RCD learns a correct maximally oriented model $\mathcal{M}$ assuming perfect conditional independence tests, sufficient hop threshold h, and sufficient depth.*

Theorem 3.3.1 states that RCD is sound and complete assuming perfect conditional independence tests. Replacing the conditional independence test by an oracle function able to provide dependencies derived from applying relational d-separation (cf. Definition 3.2.24) allows to give a perfect canonical dependencies list.

3.3.3 Relational hybrid approaches

Despite the panoply of works presented for standard BNs structure learning, only few works have been devoted to learn their relational extension. As we have seen in section 1.4.2, hybrid approaches use both statistical conditional independence tests and a global structure learning approach. The relational extension of the greedy search algorithm has been well developed. However, constraint-based approaches based on the use of statistical tests are under study. Even the existing approaches present some deficiencies when applying statistical tests due to the asymmetry caused by the use of aggregate functions. To the best of our knowledge, there is no work in this direction. Chapter 5 of this manuscript is dedicated to a deeper discussion about performing conditional independence tests in a relational context. In this chapter we also present a new hybrid approach to learn RBNs structure from a complete relational database instance.

3.4 Evaluating relational Bayesian networks structure learning algorithms

As seen in Section 3.3, only few works have been proposed to learn relational BNs extensions from relational data. As for standard BNs, evaluating the effectiveness of the proposed approaches is needed. The evaluation process goes through three main steps as depicted by Figure 3.12. 1) Generating relational

database instance from an already known RBN or from a generated one. 2) Applying the learning approach to the generated data. 3) Comparing the learned network with the gold one using some evaluation metrics. In this section we discuss existing approaches to generate synthetic RBNs and relational observational data. Then we focus on relational evaluation metrics.

3.4.1 Random generation of relational Bayesian networks

Random probabilistic relational models generation has to be established in order to evaluate proposed learning approaches in a common framework. (Maier et al., 2010) used a predefined schema and have only generated a number of dependencies varying from 5 to 15 and the conditional probability tables for attributes from a Dirichlet distribution. In (Maier et al., 2013a) the authors have generated relational synthetic data to perform experimentation. Their generation process is based only on a particular family of relational schemas, with N classes (nodes) and $N-1$ referential constraints (edges). Whereas in real world cases, relational schemas may have more than $N-1$ referential constraints. If the schema is fully connected (as described in (Maier et al., 2013c)), it will follows a tree structure. Torti et al. (Torti et al., 2010) proposed a slightly different representation of RBNs, developed in the basis of the object-oriented framework and expert knowledge. Their main issue was probabilistic inference rather than learning. In their experimental studies, (Wuillemini and Torti, 2012) have randomly generated RBNs using the layer pattern. The use of this architecture pattern imposes a particular order when searching for connections between classes, generating reference slots of the relational schema and also when creating the relational skeleton. No indication has been made about the generation of probabilistic dependencies between attributes. In addition, they are not interested neither in populating a relational database nor in communicating with a database management system. In chapter 4, we turn back to this issue and we present a new approach to generate relational Bayesian networks from scratch using a broader range of relational schemas.

3.4.2 Sampling relational Bayesian networks

This process is equivalent to generating data from a Bayesian network. Having a theoretical RBN and an instantiation of this model,we can generate as many relational database instances as needed by sampling from the constructed GBN.

3.4.3 Evaluation metrics

As we have discussed in Section 1.5.3), evaluation metrics of BN structure learning algorithms can concern either the execution speed of the proposed algorithm or the quality of the graph reconstruction. As RBNs structure learning algorithms are inspired from BNs structure learning ones, it would be natural to perform same strategies as for BNs when evaluating those proposals. Evaluation with respect to the execution time will further depend on the used DBMS quality. Evaluation with respect to the number of statistical calls performed by an algorithm will be exactly the same as for BNs. Evaluation with respect to the quality of reconstruction needs to be adapted. Methods based on sensitivity and specificity as well as those based on score functions may be applied as they are whereas, they will preserve same drawbacks already discussed in section 1.5.3. Methods based on distance measures give better tool to compare theoretical dependency structure with the learned one but they have to be adapted to the relational data representation. Further discussion about those methods will be given in Chapter 4.

In (Maier et al., 2013a), the authors have used the precision, recall and F-score measures to evaluate the quality of the learned graph structure:

Precision: The ratio of the number of relevant edges retrieved to the total number of relevant and irrelevant edges retrieved in the learned PRM dependency structure $\mathcal{S}_{Learned}$. Relevant edges are those that are present in the true model.

$$\text{Precision} = \frac{\text{Number of relevant edges retrieved in } \mathcal{S}_{Learned}}{\text{Number of edges in } \mathcal{S}_{Learned}} \tag{3.8}$$

Recall: The ratio of the number of relevant edges retrieved to the total number of relevant edges int the true PRM dependency structure $\mathcal{S}_{True}$, which is generated using the random generation process.

$$\text{Recall} = \frac{\text{Number of relevant edges retrieved } \mathcal{S}_{Learned}}{\text{Number of edges in } \mathcal{S}_{True}} \tag{3.9}$$

F-score: The F-measure is used to provide a weighted average of both precision and recall.

$$\text{F-score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.10}$$

Due to the reduced number of learning approaches, comparison with existing approaches and the evaluation task are so far not yet discussed.

On the other hand, we note that there is no adaptation of performance measures used to evaluate BN Learning algorithms (see Section) to the relational context. This is due to the fact that a majority of these measures use essential graphs when comparing the learned graph to the true one. However, this notion is not yet defined in the relational context and an adaptation of the structure learning evaluation metrics requires foremost the definition of Markov equivalence in a relational context.

3.5 Relational Bayesian network-related softwares

Some software tools have been already provided to address different challenging tasks related to probabilistic relational models in general. A non exhaustive list contains:

ProbReM. [3] Provides inference methods in DAPER models (Heckerman et al., 2004).

Alchemy. [4] Presents a list of algorithms for statistical relational learning and probabilistic logic inference in Markov logic networks.

Primula. [5] A software tool allowing to perform inference and parameter learning for RBNs of Jaeger (cf. Section 3.2).

BLOG. [6] A programming language allowing to deal with structural uncertainty in Relational modeling.

Unbbayes. [7] Limited to simple RBN models representation. It does not provide a tool to learn them.

Proximity. [8] This tool allows to work with several relational models (e.g.,relational Bayesian classifier (Neville et al., 2003b), relational probability trees (Neville et al., 2003a), Relational dependency networks (Neville and Jensen, 2007)) and provides algorithms to learn them from relational data. Yet Proximity does not provide such functionalities for RBNs.

A GRaphical Universal Model (AGrUM). [9] This C++ library is designed to work with various graphical models either propositional or relational. As relational functionalities, the library allows to specify

3. http://www.cs.mcgill.ca/~fkaeli/probrem/index.html
4. http://alchemy.cs.washington.edu/
5. http://people.cs.aau.dk/~jaeger/Primula/
6. http://bayesianlogic.github.io/
7. http://unbbayes.sourceforge.net/
8. https://kdl.cs.umass.edu/display/public/Proximity
9. https://forge.lip6.fr/projects/agrum

a special RBN representation that is based on the object-oriented paradigm (Wuillemini and Torti, 2012). In addition it provides algorithms to perform lifted probabilistic inference for this model representation. A Python version of this library has been provided, called **pyAgrum** (https://forge.lip6.fr/projects/pyagrum).

On one hand, these tools deal with a variety of relational probabilistic languages. A comparative study of a subset of these systems has been started by Manfred Jaege [10]. On the other hand, we notice the lack of works on the context of learning relational Bayesian network structure. No implementation has been found for the RGS algorithm. For RPC and RCD the code is made public. However, they are built on the DAPER specification. Consequently, we have developed a new platform to deal with RBNs. Details about our software tool are provided by Chapter 6.

3.6 Conclusion

In this chapter we have defined relational Bayesian networks and other similar relational representations. Then we have provided a survey on existing approaches for learning relational models structure. Only few methods are presented in the literature while the formalism has been developed since the early 2000s. This can be argued, on one hand, by the complexity of this formalism although it is more realistic and expressive than conventional models based on the flat representation. On the other hand, by the absence of a counterpart in the relational domain of some basic existing theoretical concepts for probabilistic graphical models. We have also shown that we lack of famous RBNs as well as random process generation.

In the next part of this manuscript, we will move on our contributions: In Chapter 4, we will address the random generation of relational Bayesian networks as well as evaluation measures. In Chapter 5, we will provide a new hybrid approach to learn RBN structure from relational data. In Chapter 6, we will provide details about the implementation choice and the main components of our platform. Finally, in Chapter 7, we will consolidate our theoretical contributions with an empirical study.

10. http://people.cs.aau.dk/ jaeger/plsystems/index.html

Propositions

4

RBN Benchmark generation and learning evaluation

Even though a panoply of works has focused, separately, on Bayesian Networks and relational databases random generation, no work has been identified for RBNs on that track. This chapter presents our first contribution. We propose an algorithmic approach allowing to generate random RBNs from scratch, then populate a relational database. The originality of this process is that it allows to generate synthetic relational data from a randomly generated relational schema and a random set of probabilistic dependencies. This process is imperative for statistical relational learning researchers to evaluate the effectiveness of their learning approaches. On the other hand, it can be of interest for database designers to be used as a decision support benchmark. It allows to generate various relational schemas, from simple to complex ones, and to populate database tables with huge number of tuples derived from distributions defined by the generated RBN. Also, we propose a new distance-based evaluation metric able to compare two relational models, one with respect to the other. This metric is an extension of the structural hamming distance SHD to the relational context.

Contents

4.1 Introduction

The evaluation process of a RBN structure learning algorithm, in the same way than BNs, requires the existence of a theoretical (also called gold) network from which one can sample a training data. Then learning is performed using this sampled data and the learned network is compared to the theoretical one using some evaluation metrics. This process needs either real known networks or randomly generated ones. However, contrary to Bayesian network, neither the first nor the second are available. Unfortunately, already existing database benchmarks are not relevant in our context. In fact, these benchmarks allow to generate databases with respect to the functional constraints present in the relational schema, whereas our requirement is quite different. Using this data, we can construct a RBN structure following one structure learning approach, but we will not be able to judge the quality of the used learning algorithm as this data is not derived from an already existing RBN. Consequently, we are in need of a random generation process of synthetic RBNs from which we can sample datasets. The relational databases generated following this process are derived from distributions defined by the generated RBNs.

The first part of the chapter proposes a benchmark random generation process. As we are working with a relational variety of Bayesian networks, our generation process will be inspired from classical methods of random generation of BNs while respecting the relational domain representation. The second part of the chapter concerns the evaluation metrics. In order to complete the evaluation process, we propose a new distance-based evaluation metric that we refer to as Relational Structural Hamming Distance RSHD for short. This distance is an adaptation of the SHD metric (cf. 1.5) to the relational context.

Section 4.2 explains the principle of our benchmark random generation process and details it. Section 4.3 presents the relational extension of the SHD metric.

The remainder of this chapter is as follows: Section 4.2 explains the principle of our contribution and details it. Section 4.2.8 provides a toy example that illustrates all the steps of our benchmark generation process from the random generation of PRM and database to their population. Section 4.3 presents the relational extension of the SHD metric.

4.2 RBN Benchmark Generation

Due to the lack of famous RBNs in the literature, this section proposes a synthetic approach to randomly generate probabilistic relational models from scratch and to randomly instantiate them and populate relational databases. To the best of our knowledge, this full process has not yet been addressed.

4.2.1 Principle

As we are working with a relational variety of Bayesian networks, our generation process will be inspired from classical methods of random generation of BNs (cf. Section 1.5) while respecting the relational domain representation.

The overall process is outlined in Algorithm 16 and illustrated by Figure 4.1. Roughly, the proposed generation process is divided into three main steps:

- The first step generates both the relational schema and the graph dependency structure using $Generate\_Relational\_Schema$, $Generate\_Dependency\_Structure$ and $Determinate\_Slot\_Chains$ functions respectively (Sections 4.2.2 and 4.2.3). Then the conditional probability tables are generated by the $Generate\_CPD$ function in the same way than Bayesian networks (cf. Section 1.4).
- The second step instantiates the model generated in the first step by generating the relational skeleton using the $Generate\_Relational\_Skeleton$ function (Section 4.2.4). The $Create\_GBN$ function creates the GBN, from both the generated RBN and the generated relational skeleton, following the steps described in Section 3.2.1.
- The third step presents the $Sampling$ function. It consists of a database instance population and it may be performed using a standard sampling method over the GBN (Section 4.2.5).

Algorithm 16 RandomizeRBN-DB

Require: N: the number of relations, $\mathcal{K}_{max}$: The maximum slot chain length allowed
Ensure: Π: $< \mathcal{R}, \mathcal{S}, CPD >$, $DB\_Instance$

Step 1: Generate the RBN
$\Pi.\mathcal{R} \leftarrow Generate\_Relational\_Schema(N)$
$\Pi.\mathcal{S} \leftarrow Generate\_Dependency\_Structure(\Pi.\mathcal{R})$
$\Pi.\mathcal{S} \leftarrow Determinate\_Slot\_Chains(\Pi.\mathcal{R}, \Pi.\mathcal{S}, \mathcal{K}_{max})$
$\Pi.CPD \leftarrow Generate\_CPD(\Pi.\mathcal{S})$
Step 2: Instantiate the RBN
$\sigma_r \leftarrow Generate\_Relational\_Skeleton(\Pi.\mathcal{R})$
$GBN \leftarrow Create\_GBN(\Pi, \sigma_r)$
Step 3: Database population
$DB\_Instance \leftarrow Sampling(GBN)$

4.2.2 Relational schema random generation

The relational schema generation process is depicted by Algorithm 17. Our aim is to generate, for a given number of classes (relations) N, a relational schema, with respect to the relational model definition presented in section 2.3.1 and where generated constraints allow to avoid referential cycles. We apply elements from the graph theory for random schema generation. We associate this issue to a DAG structure generation process, where nodes represent relations and edges represent referential constraints definition. $X_i \rightarrow X_j$ means that X_i is the referencing relation and X_j is the referenced one. Besides, we aim to construct schemas where $\forall\{X_i, X_j\} \in \mathcal{X}$ there exist a referential path from X_i to X_j. This assumption allows to browse all classes in order to discover probabilistic dependencies later and it is translated by searching DAG structures containing a single connected component (i.e., connected DAG).

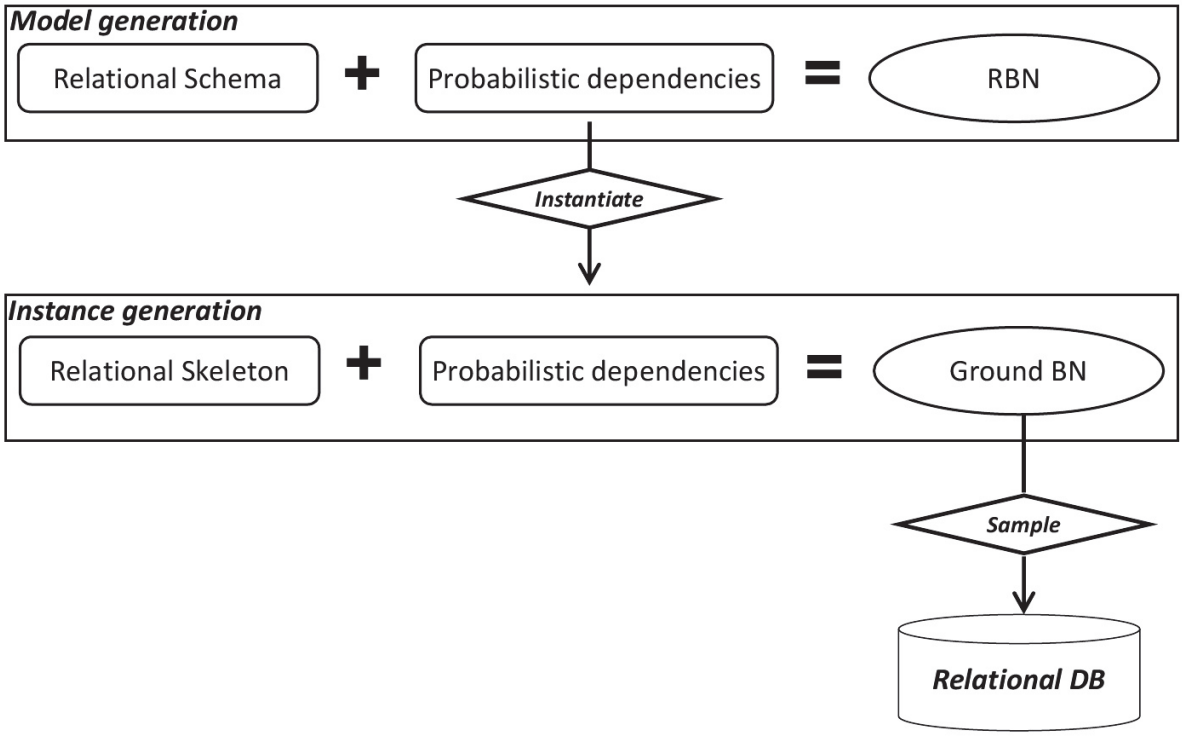


Figure 4.1: Overview of the generation and population process

Algorithm 17 *Generate_Relational_Schema*

Require: N: the number of classes
Ensure: $\mathcal{R}$: The generated relational schema
1: **repeat**
2: $\mathcal{G}$ is a connected DAG
3: **until** $\mathcal{G} \leftarrow Generate\_DAG(Policy)$
4: **for** each relation $X_i \in \mathcal{R}$ **do**
5: $\mathcal{P}k\_X_i \leftarrow Generate\_Primary\_Key(X_i)$
6: $\mathcal{A}(X_i) \leftarrow Generate\_Attributes(Policy)$
7: $\mathcal{V}(X_i.A) \leftarrow Generate\_States(Policy)$
8: **end for**
9: **for** each $n_i \rightarrow n_j \in \mathcal{G}$ **do**
10: $Fk\_X_i \leftarrow Generate\_Foreign\_Key(X_i, X_j, Pk\_X_j)$
11: **end for**

Having a fixed number of relations N, the $Generate\_DAG$ function constructs a DAG structure $\mathcal{G}$ with N nodes, where each node $n_i \in \mathcal{G}$ corresponds to a relation $X_i \in \mathcal{R}$ following various possible implementation policies (cf. Section 4.2.6). For each class we randomly generate a primary key attribute using the $Generate\_Primary\_Key$ function. Then, we randomly generate the number of attributes and their associated domains using the $Generate\_Attributes$ and $Generate\_States$ functions respectively. Note that the generated domains do not take into account possible probabilistic dependencies between attributes. For each $n_i \rightarrow n_j \in \mathcal{G}$, we generate a foreign key attribute in X_i using the $Generate\_Foreign\_Key$ function.

4.2.3 RBN random generation

Generated schemas are not sufficient to generate database instances where the attributes are not independent. We need to randomly generate probabilistic dependencies between the attributes of the schema classes. These dependencies have to provide the DAG of the dependency structure $\mathcal{S}$ and a set of CPDs which define a RBN (cf. definition 3.2.3).

We especially focus on the random generation of the dependency structure. Once this latter is identified, conditional probability distributions may be sampled in a similar way as standard BNs parameter generation.

The dependency structure $\mathcal{S}$ should be a DAG to guarantee that each generated ground network is also a DAG (Getoor, 2002). $\mathcal{S}$ has the specificity that one descriptive attribute may be connected to another with different possible slot chains. Theoretically, the number of slot chains may be infinite. In practice a user-defined maximum slot chain length K_{max}, is specified to identify the horizon of all possible slot chains. In addition, the K_{max} value should be at least equal to $N - 1$ in order to not neglect potential dependencies between attributes of classes connected via a long path. Each edge in the DAG has to be annotated to express from which slot chain this dependency is detected. We add dependencies following two steps. First we add oriented edges to the dependency structure while keeping a DAG structure. Then we identify the variable from which the dependency has been drawn by a random choice of a legal slot chain related to this dependency.

4.2.3.1 Constructing the DAG structure

The DAG structure identification is depicted by Algorithm 18. The idea here is to find for each node $X.A$ a set of parents from the same class or from further classes while promoting intra-class dependencies in order to control the final model complexity as discussed in (Getoor, 2002). This condition promotes the discovery of intraclass dependencies or those coming from short slot chains. The more the chain slot is long, the less a probabilistic dependency through this slot chain may be found. To follow this condition, having

Algorithm 18 Generate_Dependency_Structure

Require: $\mathcal{R}$: The relational schema
Ensure: $\mathcal{S}$: The generated relational dependency structure
1: **for** each class $X_i \in \mathcal{R}$ **do**
2: $\mathcal{G}_i \leftarrow Generate\_Sub\_DAG(Policy)$
3: **end for**
4: $\mathcal{S} \leftarrow \bigcup \mathcal{G}_i$
5: $\mathcal{S} \leftarrow Generate\_Super\_DAG(Policy)$

Algorithm 19 Determinate_Slot_Chains

Require: $\mathcal{R}$: The relational schema, $\mathcal{S}$: The dependency structure, K_{max}: The maximum slot chain length
Ensure: $\mathcal{S}$: The generated relational dependency structure with generated slot chains
1: $K_{max} \leftarrow max(K_{max}, card(\mathcal{X}_{\mathcal{R}}) - 1)$
2: **for** each $X.A \rightarrow Y.B \in \mathcal{S}$ **do**
3: $Pot\_Slot\_Chains\_List \leftarrow Generate\_Potential\_Slot\_chains(X, Y, \mathcal{R}, K_{max})$
4: **for** each $slot\_Chain \in Pot\_Slot\_Chains\_List$ **do**
5: $l \leftarrow length(slot\_Chain)$
6: $W[i] \leftarrow \exp^{\frac{-l}{nb\_Occ(l, Pot\_Slot\_Chains\_List)}}$
7: **end for**
8: $Slot\_Chain^* \leftarrow Draw(Pot\_Slot\_Chains\_List, W)$
9: **if** $Needs\_Aggregator(Slot\_Chain^*)$ **then**
10: $\gamma \leftarrow Random\_Choice\_Agg(list\_Aggregators)$
11: **end if**
12: **if** $Slot\_Chain^* = 0$ **then**
13: $\mathcal{S}.Pa(X.A) \leftarrow \mathcal{S}.Pa(X.A) \cup Y.B$ % *here* $X = Y$
14: **else**
15: $\mathcal{S}.Pa(X.A) \leftarrow \mathcal{S}.Pa(X.A) \cup \gamma(Y.Slot\_Chain^*.B)$
16: **end if**
17: **end for**

N classes, we propose to construct N separated sub-DAGs, each of which is built over attributes of its corresponding class using the $Generate\_Sub\_DAG$ function. Then, we construct a super-DAG over all the previously constructed sub-DAGs. At this stage, the super-DAG contains N disconnected components: The idea is to add inter-classes dependencies in such a manner that we connect these disconnected components while keeping a global DAG structure.

To add inter-class dependencies we constrain the choice of adding dependencies among only variables that do not belong to the same class. For an attribute $X.A$, the $Generate\_Super\_DAG$ function chooses randomly an attribute $Y.B$, where $X \neq Y$, then verifies whether the super-DAG structure augmented by a new dependency from $X.A$ to $Y.B$ remains a DAG. If yes it keeps the dependency otherwise it rejects it and searches for a new one. Used policies are discussed in Section 4.2.6.

4.2.3.2 Determining slot chains

During this step, we have to take into consideration that one variable may be reached through different slot chains and the dependency between two descriptive attributes will depend on the chosen one. The choice has to be made randomly while penalizing long slot chains. We penalize long indirect slot chains, by having the probability of occurrence of a probabilistic dependence from a slot chain length l proportional to $\exp^{-l}$ (Getoor, 2002). Having a dependency $X.A \rightarrow Y.B$ between two descriptive attributes $X.A$ and $Y.B$, we start by generating the list of all possible slot chains ($Pot\_Slot\_Chains\_List$) of $length \leq K_{max}$

Algorithm 20 Generate_Relational_Skeleton

Require: $\mathcal{R}$: The relational schema
Ensure: σ_r: The generated relational skeleton
1: **for** each class $\rho \in \mathcal{R}$ **do**
2: $nb\_Objects_{\rho.referencing} \leftarrow Draw\_Objects(policy)$
3: $nb\_Objects_{\rho.referenced} \leftarrow Draw\_Objects(policy)$
4: $\mathcal{O}^{\rho.referenced} \leftarrow Generate\_Objects(nb\_Objects_{\rho.referenced}, \rho.referenced)$
5: $\mathcal{O}^{\rho.referencing} \leftarrow Generate\_Objects(nb\_Objects_{\rho.referencing}, \rho.referencing)$
6: $nb\_Related \leftarrow Draw\_Objects(policy)$
7: $\sigma_r \leftarrow Add\_Links(\mathcal{O}^{\rho.referenced}, \mathcal{O}^{\rho.referencing}, nb\_Related)$
8: **end for**

from which X can reach Y in the relational schema using the $Generate\_Potential\_Slot\_chains$ function. Then, we create a vector W of the probability of occurrence for each of the found slot chains, with $\log(W[i]) \propto \frac{-l}{nb\_Occ(l,Pot\_Slot\_Chains\_List)}$, where l is the slot chain length and nb_Occ is the number of slot chains of length $l \in Pot\_Slot\_Chains\_List$. This value will rapidly decrease when the value of l increases which allows to reduce the probability of selecting long slot chains. We then sample a slot chain from $Pot\_Slot\_Chains\_List$ following W using the $Draw$ function. If the chosen slot chain implies an aggregator, then we choose it randomly from the list of existing ones using the $Random\_Choice\_Agg$ function. The slot chain determination is depicted by Algorithm 19.

Following our approach, database population requires the instantiation of the previously generated RBN. Both steps are detailed below.

4.2.4 GBN generation

The generated schema together with the added probabilistic dependencies and generated parameters give rise to the probabilistic relational model. To instantiate this latter, we need to generate a relational skeleton by generating a random number of objects per class, then adding links between objects. This step is strongly related to the reference slot notion. That is, all referencing classes have their generated objects related to objects from referenced classes. In Algorithm 20, we start by generating the number of objects of referencing and referenced classes using the $Draw\_Objects$ function. Then we generate the objects, referencing and referenced ones, by randomly instantiating their corresponding classes using the $Generate\_Objects$ function. We specify the number of related ones using the $Draw\_Objects$ function and finally, we relate them using the Add_Links function. Used policies are discussed in Section 4.2.6.

The GBN is fully determined with this relational skeleton and the CPDs already present at the meta-level.

4.2.5 Database population

This process is equivalent to generating data from a Bayesian network. We can generate as many relational database instances as needed by sampling from the constructed GBN. The specificity of the generated tuples is that they are sampled not only from functional dependencies but also from probabilistic dependencies provided by the randomly generated RBN.

4.2.6 Implemented policies for the generation process

Policy for generating the relational schema DAG structure. To randomly generate the relational schema DAG structure, we use the PMMixed algorithm (cf. Section 1.3). This latter leads to generate uniformly distributed DAGs in the DAGs space. Consequently the generated structure may be a disconnected

graph yet, we need a DAG structure containing a single connected component. To preserve this condition together with the interest of generating uniformly distributed examples, we follow the *rejection sampling technique*. The idea is to generate a DAG following the PMMixed principle, if this DAG contains just one connected component, then it is accepted, otherwise it is rejected. We repeat these steps until generating a DAG structure satisfying our condition.

Policies for generating attributes and their cardinalities. Having the graphical structure, we continue by generating, for each relation R, a primary key attribute, a set of attributes $\mathcal{A}$, where $card(\mathcal{A}) - 1 \sim Poisson(\lambda = 1)$, to avoid empty sets, and for each attribute $A \in \mathcal{A}$, we specify a set of possible states $\mathcal{V}(A)$, where $card(\mathcal{V}(A)) - 2 \sim Poisson(\lambda = 1)$.

Policies for generating the dependency structure. We follow the PMMixed algorithm principle to construct a DAG structure inside each class. Then, to add inter-class dependencies we use a modified version of the PMMixed algorithm where we constrain the choice of adding dependencies among only variables that do not belong to the same class.

Policy for generating the relational skeleton. The number of generated objects either for the referenced or the referencing classes as well as the number of interlinked objects $\sim Poisson(\lambda = N1)$ and $\sim Poisson(\lambda = N2)$ respectively. $N1$ and $N2$ are user-defined.

4.2.7 Time complexity of the generation process

Time complexity of the random generation process is closely related to the choice of the implementation policies. Let N be the number of relations (classes), we report the average complexity of each step of the generation process.

Complexity of the relational schema generation process. Algorithm 17 is structured of three loops. Namely, the most expensive one is the first loop dedicated for the DAG structure construction and uses the PMMixed algorithm. Time complexity of the PMMixed algorithm is $\mathcal{O}(N * \lg N)$. This algorithm is called until reaching the stop condition (i.e., a connected DAG). Let T be the average number of calls of the PMMixed algorithm. T is the ratio of the number of all connected DAG constructed from N nodes (Robinson, 1977) to the number of all DAGs constructed from N nodes (Bender and Robinson, 1988). Time complexity of Algorithm 17 is $\mathcal{O}(T * N * \lg N)$.

Complexity of the dependency structure generation process. As for Algorithm 17, the most expensive operation of Algorithm 18 is the generation of the DAG structure inside each class $X_{i \in \{1 \ldots N\}} \in \mathcal{X}$. Through Algorithm 17, a set of attributes $\mathcal{A}(X_i)$ has been generated for each X_i. As $card(\mathcal{A}(X_i)) - 1 \sim Poisson(\lambda = 1)$, following Section 4.2.6, Then the average number of generated attributes for each class is $lambda = 1 + 1 = 2$. Then time complexity of the algorithm is $\mathcal{O}(N * 2 * \lg 2)$.

Complexity of the slot chains determination process. The most expensive operation of Algorithm 19 is the $Generate\_Potential\_Slot\_chains$ method. This latter explore recursively the relational schema graph in order to find all paths (i.e., slot chains) of length $k \in \{0 \ldots K_{max}\}$. Time complexity of this method is $\mathcal{O}(N^{K_{max}})$.

Complexity of the relational skeleton generation process. In Algorithm 20, the $generate\_Objects$ method allows to generate a random number of objects per class. The average number of generated objects per class $\sim Poisson(\lambda = N1) = N1$. The average number of attributes for each object is equal to the average number of attributes at the class level which is equal to 2. Let E be the number of $\rho \in \mathcal{R}$, then time complexity of this method is $\mathcal{O}(N1 * 2 * E)$.

4.2.8 Toy example

In this section, we illustrate the benchmark random generation process through a toy example.

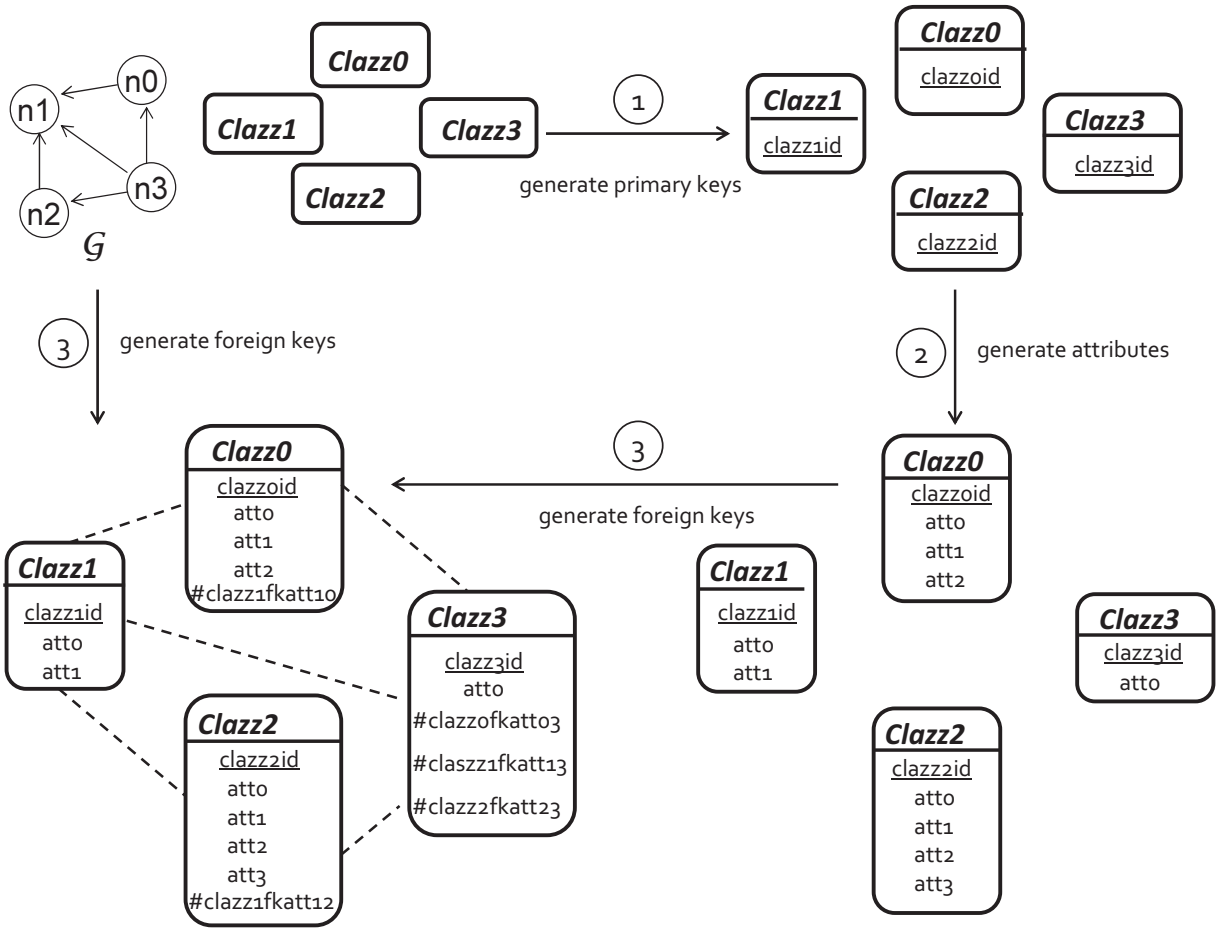


Figure 4.2: Relational schema generation steps

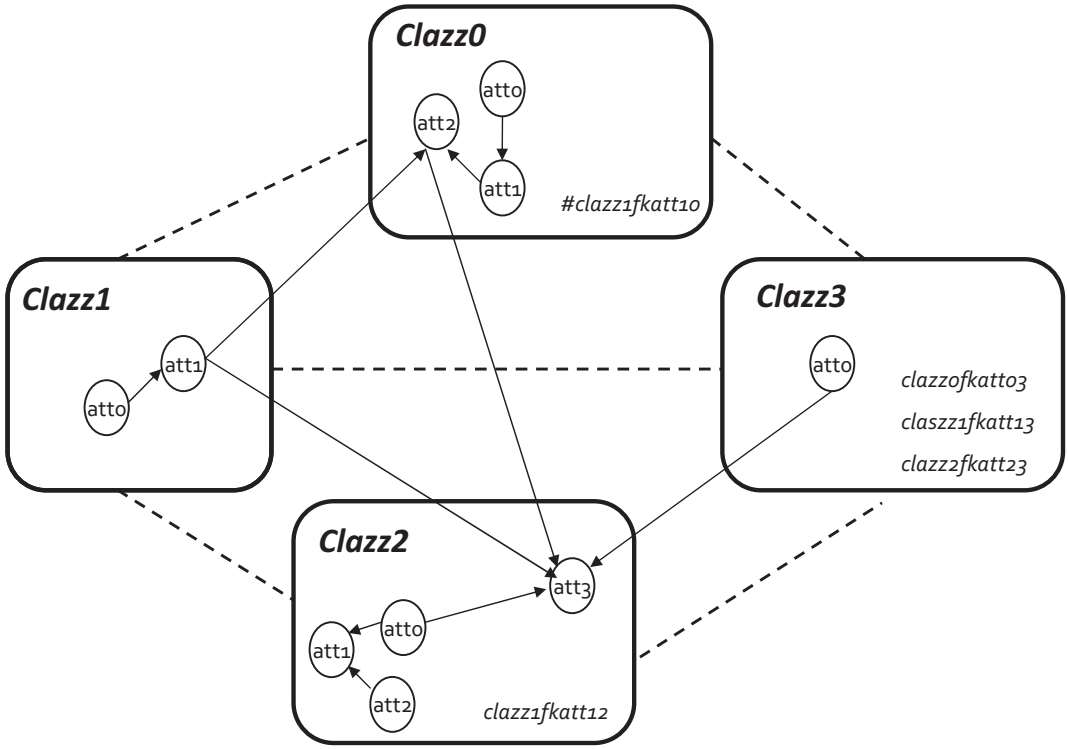


Figure 4.3: Graph dependency structure generation

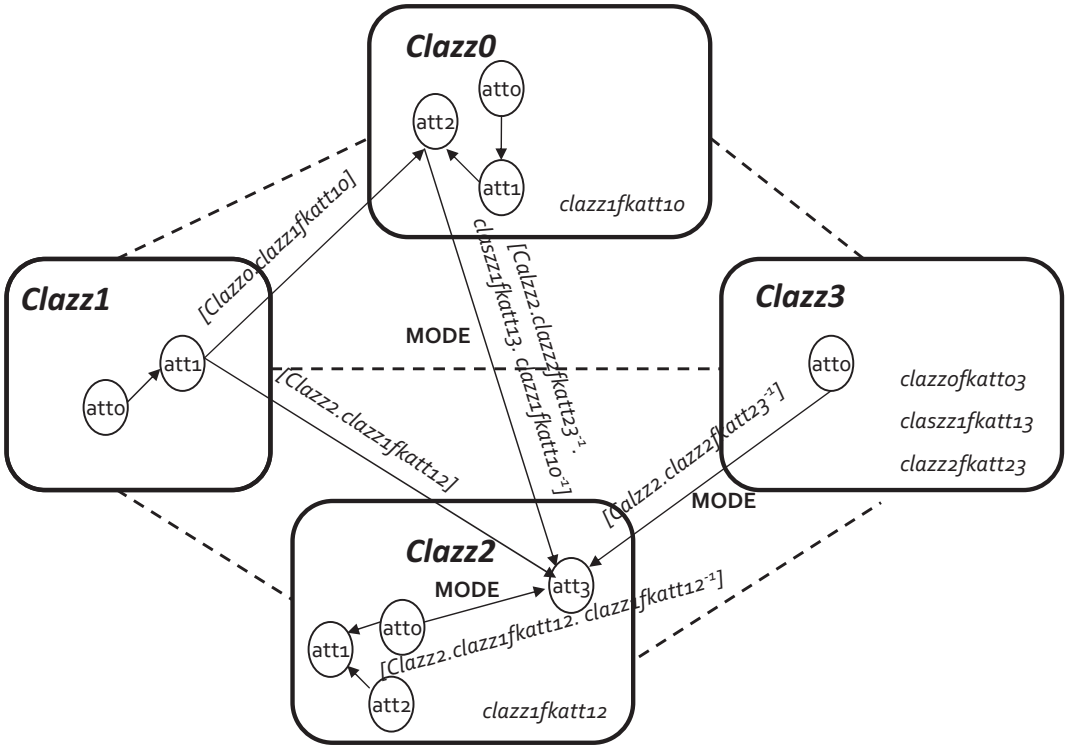
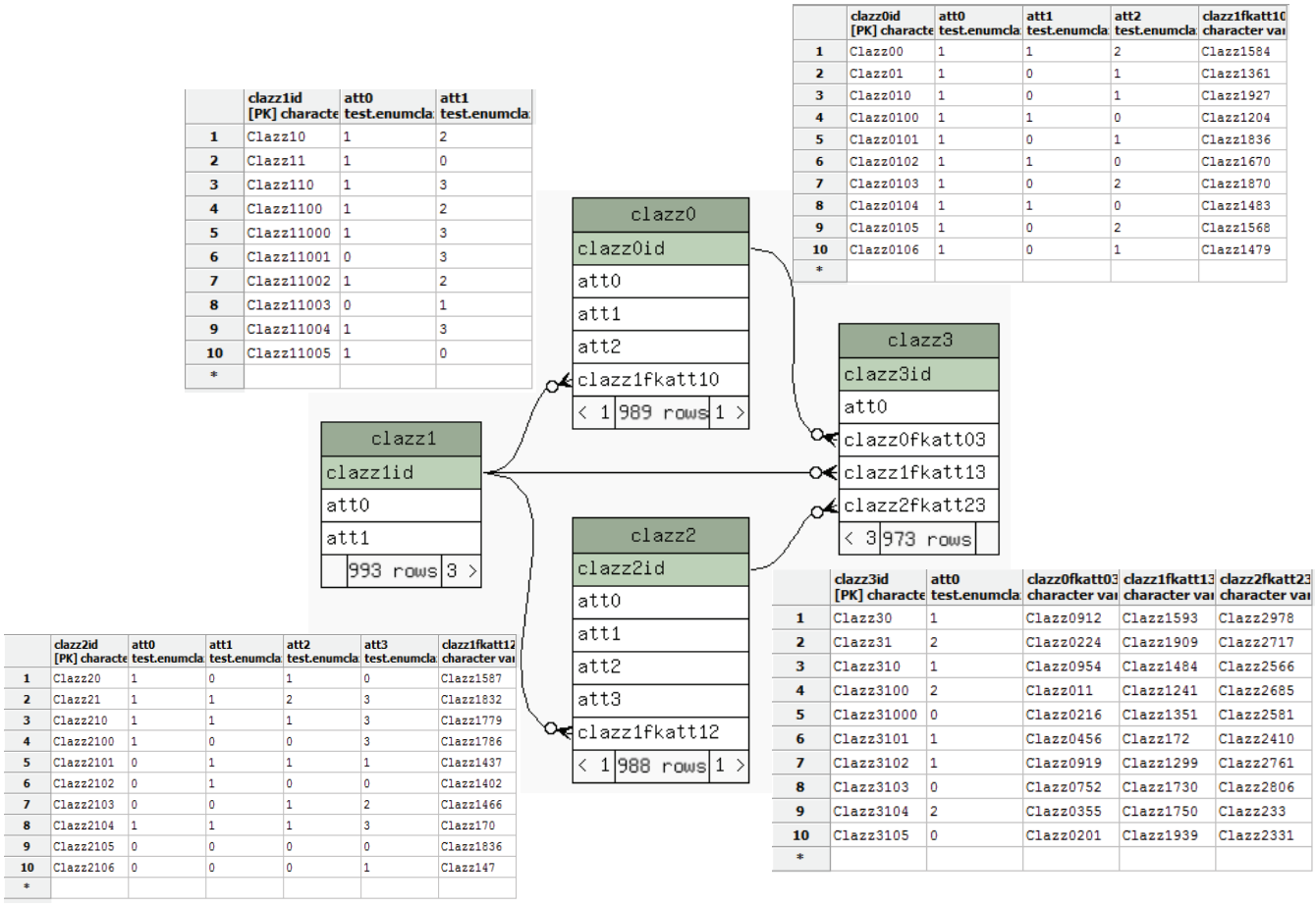


Figure 4.4: Example of a generated relational schema where the dotted lines represent referential constraints and the generated RBN dependency structure where the arrows represent probabilistic dependencies. we omit to specify slot chains to not overload the figure. Details about slot chains from which probabilistic dependencies have been detected are given in Paragraph *RBN generation*.

Relational schema generation. Figure 4.2 presents the result of running Algorithm 17, with $N = 4$ classes. For each class, a primary key has been added ($clazz0id$, $clazz1id$, $clazz2id$ and $clazz3id$). Then a number of attributes have been generated randomly together with a set of possible states for each attribute using the policies described in Section 4.2.6 (e.g., $clazz0$ has 3 descriptive attributes $att0, att1$ and $att2$. $att0$ is a binary variable). Finally, foreign key attributes have been specified following the DAG structure of the graph $\mathcal{G}$ (e.g., $clazz2$ references class $clazz1$ using foreign key attribute $clazz1fkatt12$).

RBN generation. Having the relational schema of the previous step, we will randomly generate a RBN. We recall that this process consists of two steps: randomly generate the dependency structure $\mathcal{S}$ (Algorithm 18), then randomly generate the conditional probability distributions which is similar to parameter generation of a standard BN. The random generation of $\mathcal{S}$ is performed in two phases. We start by constructing the DAG structure, the result of this phase is in Figure 4.3. Then, we fix a maximum slot chain length K_{max} to randomly determine from which slot chain the dependency has been detected. We use $K_{max} = 3$, the result of this phase gives rise to the graph dependency structure of Figure 4.4. $\mathcal{S}$ contains 5 intra-class and 5 inter-class probabilistic dependencies.
Three of the inter-class dependencies have been generated from slot chains of length 1:
$Clazz0.clazz1fkatt10.att1 \rightarrow Clazz0.att2$;
$MODE(Calzz2.clazz2fkatt23^{-1}.att0) \rightarrow Clazz2.att3$ and;
$Clazz2.clazz1fkatt12.att1 \rightarrow Clazz2.att3$
One from slot chain of length 2:
$MODE(Clazz2.clazz1fkatt12.clazz1fkatt12^{-1}.Clazz2.att0) \rightarrow Clazz2.att3$
One from slot chain of length 3:
$MODE(Calzz2.clazz2fkatt23^{-1}.claszz1fkatt13.clazz1fkatt10^{-1}) \rightarrow Clazz2.att3$



	clazz1id [PK] characte	att0 test.enumcla	att1 test.enumcla
1	Clazz10	1	2
2	Clazz11	1	0
3	Clazz110	1	3
4	Clazz1100	1	2
5	Clazz11000	1	3
6	Clazz11001	0	3
7	Clazz11002	1	2
8	Clazz11003	0	1
9	Clazz11004	1	3
10	Clazz11005	1	0
*			

	clazz0id [PK] characte	att0 test.enumcla	att1 test.enumcla	att2 test.enumcla	clazz1fkatt10 character var
1	Clazz00	1	1	2	Clazz1584
2	Clazz01	1	0	1	Clazz1361
3	Clazz010	1	0	1	Clazz1927
4	Clazz0100	1	1	0	Clazz1204
5	Clazz0101	1	0	1	Clazz1836
6	Clazz0102	1	1	0	Clazz1670
7	Clazz0103	1	0	2	Clazz1870
8	Clazz0104	1	1	0	Clazz1483
9	Clazz0105	1	0	2	Clazz1568
10	Clazz0106	1	0	1	Clazz1479
*					

	clazz2id [PK] characte	att0 test.enumcla	att1 test.enumcla	att2 test.enumcla	att3 test.enumcla	clazz1fkatt12 character var
1	Clazz20	1	0	1	0	Clazz1587
2	Clazz21	1	1	2	3	Clazz1832
3	Clazz210	1	1	1	3	Clazz1779
4	Clazz2100	1	0	0	3	Clazz1786
5	Clazz2101	0	1	1	1	Clazz1437
6	Clazz2102	0	1	0	0	Clazz1402
7	Clazz2103	0	0	1	2	Clazz1466
8	Clazz2104	1	1	1	3	Clazz170
9	Clazz2105	0	0	0	0	Clazz1836
10	Clazz2106	0	0	0	1	Clazz147
*						

	clazz3id [PK] characte	att0 test.enumcla	clazz0fkatt03 character var	clazz1fkatt13 character var	clazz2fkatt23 character var
1	Clazz30	1	Clazz0912	Clazz1593	Clazz2978
2	Clazz31	2	Clazz0224	Clazz1909	Clazz2717
3	Clazz310	1	Clazz0954	Clazz1484	Clazz2566
4	Clazz3100	2	Clazz011	Clazz1241	Clazz2685
5	Clazz31000	0	Clazz0216	Clazz1351	Clazz2581
6	Clazz3101	1	Clazz0456	Clazz172	Clazz2410
7	Clazz3102	1	Clazz0919	Clazz1299	Clazz2761
8	Clazz3103	0	Clazz0752	Clazz1730	Clazz2806
9	Clazz3104	2	Clazz0355	Clazz1750	Clazz233
10	Clazz3105	0	Clazz0201	Clazz1939	Clazz2331
*					

Figure 4.5: Visual graph representation of the generated relational schema and table records by using SchemaSpy and PostgreSQL software tools.

GBN creation. Once the RBN is generated, we follow the two steps presented in Section 4.2.4 to create a GBN and to populate the DB instance. We have generated an average number of 1000 tuple per class. The database has been stored using PostgreSQL [1] RDBMS. Figure 4.5 presents a graphical representation of the generated relational schema using SchemaSpy [2] software tool and some table records viewed from PostgreSQL interface.

After detailing and illustrating our benchmark random generation process, we turn to another interesting task, namely metrics to compare two RBNs structures. These metrics are well important when dealing with RBN structure learning approaches as they allow to assess the quality of reconstruction of a given algorithm.

4.3 Learning evaluation metrics

4.3.1 Discussion

As we have seen in Chapters 1 and 3, there are several metrics allowing to evaluate a BN structure learning approach. This concern is not yet developed for RBNs, especially when the main goal is to evaluate the quality of reconstruction. We have seen in Section 3.4.3 that Maier et al. (Maier et al., 2013a) have used the *Precision*, *Recall* and their harmonic mean, the *F-score*, to evaluate the quality of their learning algorithm. However, these measures are not well appropriate; theoretically, a perfect structure learning

1. http://www.postgresql.org/
2. http://schemaspy.sourceforge.net/

algorithm should provide a precision and a recall of value one, whereas, usually, these two requirements are often contradictory: a learning approach that returns the list of all possible dependencies will provide a 100% recall. Alternatively, we can have a high value of the precision, not because of the good quality of learning approach but because it provides a few number of learned dependencies. For instance a learning approach that is able to provide only one learned dependency given a true model with 10 dependencies will have a 100% precision, if this learned dependency is relevant, against a very low recall. Even though their harmonic mean, the F-score, allows to provide a compromise between these two measures, its value will somehow be infected by the erroneous values of the precison or the recall.

Furthermore, for relational dependency structures, a dependency is defined among three components, namely, the presence of the edge, the slot chain from which the dependency is constructed and the used aggregator. (Maier et al., 2013a) have computed the Precision, Recall and F-score to only penalize missing edges, extra edges, and reversed edges and they omit slot chains and (or) aggregators.

4.3.2 Penalization for relational models

In this section, we introduce two new terms of penalization, the first penalizes wrong slot chains while the second penalizes poorly learned aggregator.

- **Penalization for wrong slot chains.** Let $\alpha \in [0, 1]$ be the term of penalization to be used to express the fact that a dependency may be discovered between two nodes but it is not derived from the right slot chain in the gold model structure. α is weighted by the longest common slot chain from the slot chains of the true and learned dependencies and is expressed as follows:

$$\alpha = 1 - \frac{length(Max\_Common\_sub\_Slot(\text{true dependency, learned dependency}))}{\text{Max(length(true dependency),length(learned dependency))}} \tag{4.1}$$

  The $Max\_Common\_sub\_Slot$ refers to the longest sequence of slot references from the slot chains of the true dependency and the learned one. The $length(Max\_Common\_sub\_Slot)$ refers to the length of this sequence.
- **Penalization for wrong aggregators.** Let $\beta \in [0, 1]$ be the term of penalization to be used to express the fact that a dependency may be discovered between two nodes but using an aggregator different from the one used by the gold dependency. Let n_{Agg} be the number of all possible aggregators for one attribute. We define a $n_{Agg} \times n_{Agg}$ cost matrix $\mathcal{C}$ where all $\mathcal{C}(i, j) \in [0, 1]$ values are user specified costs. $\mathcal{C}(i, j) = 0$ means that the learned aggregator is the gold one.
- **Overall penalization.** To consider both slot chain and aggregator penalizations, we compute their arithmetic mean ψ defined as follows:

$$\psi = \frac{\alpha + \beta}{2} \tag{4.2}$$

4.3.3 Relational Precision and Recall

We redefine the notions of Precision and Recall presented in Section 3.4.3 as follows:

$hard\_Precision$**:** The ratio of the number of relevant dependencies retrieved to the total number of relevant and irrelevant dependencies retrieved in the learned PRM dependency structure $\mathcal{S}_{Learned}$. Relevant dependencies are those that are present in the true model: same edge, same slot chain and same aggregator.

$$\text{hard\_Precision} = \frac{\text{Number of relevant dependencies retrieved in } \mathcal{S}_{Learned}}{\text{Number of dependencies in } \mathcal{S}_{Learned}} \tag{4.3}$$

$hard\_Recall$**:** The ratio of the number of relevant dependencies retrieved to the total number of relevant dependencies in the true PRM dependency structure $\mathcal{S}_{True}$, which is generated using the random generation process.

Algorithm 21 $RSHD$

Require: $Canonic\_Deps_{gold}$: The canonical dependencies of the gold model.
$Canonic\_Deps_{learned}$: The canonical dependencies of the learned model.
Ensure: The $rshd$ value.

```
rshd = 0
for every dependency D different int Canonic_Deps_gold than Canonic_Deps_learned do
    %For missing canonical dependencies.
    if D.edge ∉ Canonic_Deps_learned then
        rshd+ = 1
    end if
    %For extra canonical dependencies.
    if D.edge ∉ Canonic_Deps_gold then
        rshd+ = 1
    end if
    %For reversed edges and edges undirected in one graph and directed in the other.
    if D.edge is incorrectly oriented in Canonic_Deps_learned then
        rshd+ = 1
    end if
    %For inappropriate discovered slot chain of the learned dependencies.
    if D.edge ∈ Canonic_Deps_gold AND D.edge ∈ Canonic_Deps_learned AND D.slot_chain is inappropriate then
        rshd+ = ψ
    end if
end for
```

$$\text{hard\_Recall} = \frac{\text{Number of relevant dependencies retrieved } \mathcal{S}_{Learned}}{\text{Number of dependencies in } \mathcal{S}_{True}} \tag{4.4}$$

The $hard\_Precision$ and $hard\_Recall$ strictly penalize wrong slot chains and aggregators. Alternatively, we can define a soft version of these metrics which takes into account the penalization terms introduced in Section 4.3.2. $soft\_Precision$ and $soft\_Recall$ are calculated as follows:

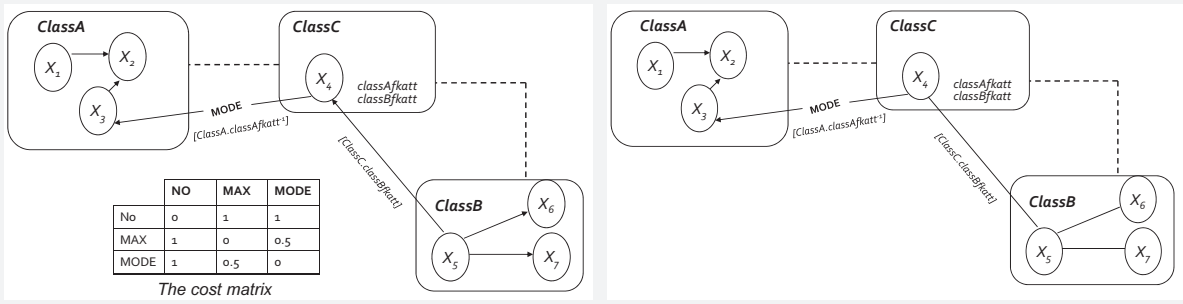
$$\text{soft\_Precision} = \frac{\sum_{i=0}^{Nb} \omega_i}{\text{Number of dependencies in } \mathcal{S}_{Learned}} \tag{4.5}$$

where Nb is the number of dependencies in $\mathcal{S}_{Learned}$ and,

$$\omega_i = \begin{cases} 1, \text{for same dependencies (i.e., same edge, slot chain and aggregator)} \\ 0, \text{for extra edges end reversed edges} \\ 1-\psi, \psi \text{defined by Formula 4.2, for relevant edges but from different slot chains/aggregators} \end{cases}$$

$$\text{soft\_Recall} = \frac{\sum_{i=0}^{Nb} \omega_i}{\text{Number of dependencies in } \mathcal{S}_{True}} \tag{4.6}$$

Even though these new definitions of Precision and Recall are more suitable to evaluate relational structure learning algorithms, they present some deficiencies. As these measures are calculated among the dependency structure, errors may be made when a difference in orientation between two dependency structures is found for symmetric dependencies (cf. Section 3.3.2). Thus, in the next section, we propose a relational distance-based measure defined on the concept of canonical dependencies presented in Section 3.3.2, rather than of the dependency structures.



(a) RBN structure #1

(b) Structure derived from the canonical dependencies of RBN structure #1

Figure 4.6: Example of a gold RBN dependency structure and its canonical dependencies

4.3.4 Relational Structural Hamming Distance

We cannot use directly the methods based on score function or on distance measures used in the context of BNs (cf. Section 1.5), and this is for many reasons:

- These metrics are based on the use of the essential graph (CPDAG), unfortunately this notion is not yet developed in the relational context. A RBN is a meta-model from which we can construct several ground graphs, having not necessarily the same equivalent class, depending on the used skeleton.
- The RBN graph structure is much more complicated than BN graphical representation due to the presence of slot chains and asymmetry caused by the use of aggregators.

Consequently, we are in need of a graphical representation similar to the essential graph representation in the relational context. The abstract ground graph (AGG) representation (cf. Section 3.2.4) cannot be considered similar as CPDAG. First, an AGG constructed from a given perspective is always fully oriented which is not the case of a CPDAG. Second an AGG allows to reason about relational d-separation from its perspective class, so it is not as general as a CPDAG.

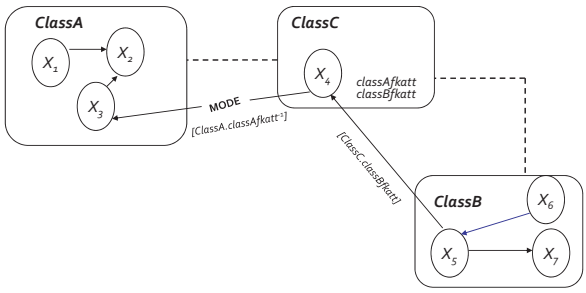
However we have seen that the RCD algorithm presented in Section 3.3.2 results on a set of canonical dependencies presenting a correct maximally oriented model $\mathcal{M}$ via Theorem 3.3.1. Moreover, we have discussed that applying RCD while substituting the conditional independence test by an 'oracle' function leads to a perfect canonical dependencies list. In this section, we will use the list of canonical dependencies derived from a learning algorithm and compare it to the list of perfect canonical dependencies derived from the true model.

Let $Canonic\_Deps_{gold}$ and $Canonic\_Deps_{learned}$ be respectively the set of canonical dependencies for both, the true and the learned structures respectively and let N be the number of classes of the RBN. Each canonical set is constructed as follows:

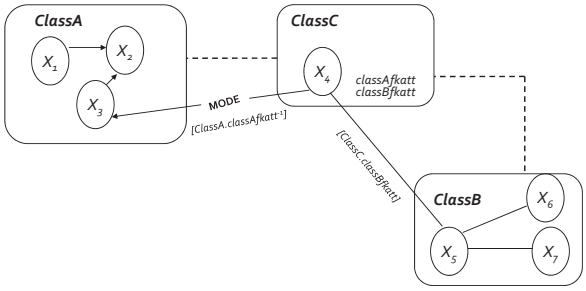
- For the true model structure, $Canonic\_Deps_{gold}$ is derived from the N abstract ground graphs (AGG) of the true model.
- For the learned model structure, $Canonic\_Deps_{learned}$ is derived from the N abstract ground graphs (AGG) of the learned model.

From these sets, we propose to define the relational structural Hamming distance, RSHD for short. As the SHD measure, $RSHD$ penalizes extra edges and missing edges by an increase of the distance by 1. When the edge exists in both structures, we turn to check whether the slot chain and aggregator of the learned dependence are those of the true dependence. If not, $RSHD$ will be incremented by a penalization of $\psi > 0$ (ψ defined by Formula 4.2). The steps of computing th RSHD measure between two sets of canonical dependencies are depicted by Algorithm 21.

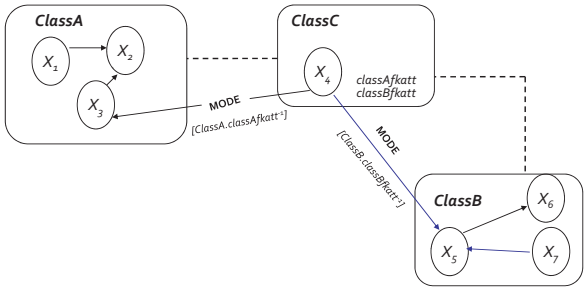
Example 4.3.1. *We provide some examples of calculating relational structural Hamming distance (RSHD). Figure 4.6 provides a hypothetical true RBN structure (Figure 4.6(a)) and the structure derived from its*



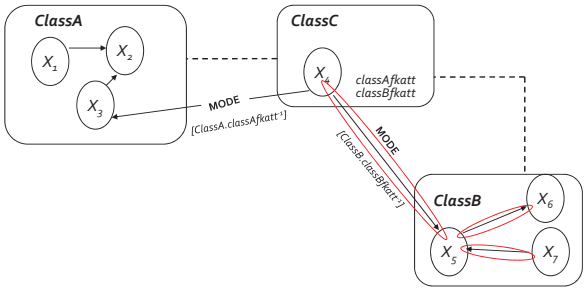
(a) RBN structure #2



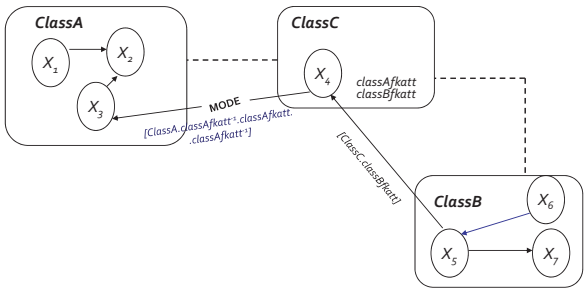
(b) Structure derived from the canonical dependencies of RBN structure #2 ($RSHD = 0$)



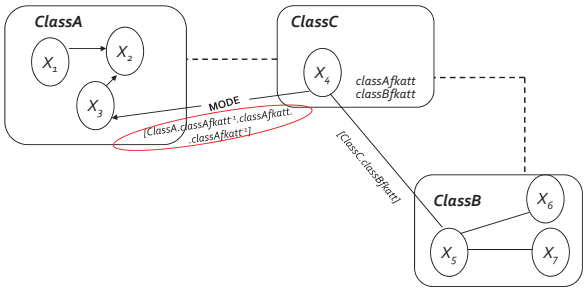
(c) RBN structure #3



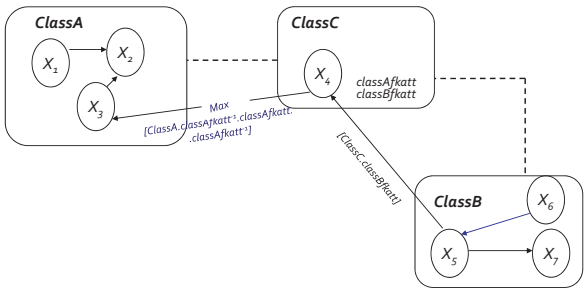
(d) Structure derived from the canonical dependencies of RBN structure #3 ($RSHD = 3$)



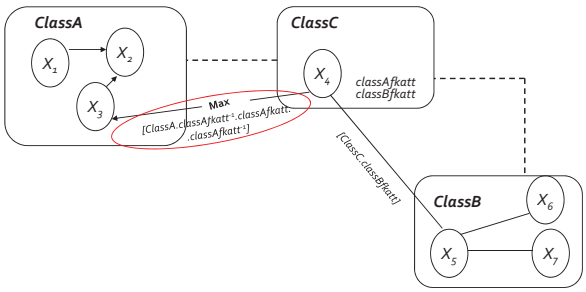
(e) RBN structure #4



(f) Structure derived from the canonical dependencies of RBN structure #4 ($RSHD = 2/3$)



(g) RBN structure #5



(h) Structure derived from the canonical dependencies of RBN structure #5 ($RSHD = (2/3 + 0.5)/2$)

Figure 4.7: Examples of calculating relational structural Hamming distance (RSHD)for the RBN of Figure 4.6

list of canonical dependencies (Figure 4.6(b)). Figures 4.7(a), 4.7(c), 4.7(e) and 4.7(g) show learned RBNs (from data sampled from the true RBN structure) and Figures 4.7(b), 4.7(d), 4.7(f) and 4.7(h) represent

respectively, the structures derived from their corresponding list of canonical dependencies. We aim to compare the canonical dependencies of Figures 4.7(b), 4.7(d), 4.7(f) and 4.7(h) respectively to the canonical dependencies of Figure 4.6(b) using the RSHD measure.

- *For the canonical dependencies of Figure 4.7(b)* $RSHD = 0$. *We notice that the RBNs of Figures 4.6(a) and 4.7(a) are different whereas they have the same set of canonical dependencies.*
- *For the canonical dependencies of Figure 4.7(d)* $RSHD = 3$. *The set of canonical dependencies of Figure 4.7(d) can match the set of canonical dependencies of Figure 4.6(b) by removing three edge orientations.*
- *For the canonical dependencies of Figure 4.7(f)* $RSHD = 2/3$. *The set of canonical dependencies of Figure 4.7(f) does not match the set of canonical dependencies of Figure 4.6(b) for only one dependency where the slot chain from which the dependency if found is not the same. This configuration is penalized by* $\alpha = 1 - \frac{1}{Max(1,3)} = 2/3$.
- *For the canonical dependencies of Figure 4.7(h)* $RSHD = (2/3 + 0.5)/2$. *The set of canonical dependencies of Figure 4.7(h) does not match the set of canonical dependencies of Figure 4.6(b) for only one dependency where both the slot chain and aggregator from which the dependency if found are not the same. This configuration is penalized by* $\psi = ((1 - \frac{1}{Max(1,3)}) + 0.5)/2$.

4.4 Conclusion

In this chapter, We have detailed our first contribution that consists on developing an algorithmic process allowing to randomly generate synthetic RBNs and instantiate them to populate a relational database. Then we have illustrated our approach using a toy example. Also, we have adapted the SHD measure to the relational context.

In the next chapter we will detail our second contribution that consists on a new approach to learn RBNs structure from relational observational data. We will highlight the utility of our generation process while comparing the new proposed structure learning approach with already existing RBN learning approaches, in a common framework, using the generated synthetic networks.

5

RMMHC: a hybrid approach to Relational Bayesian Networks structure learning

RBNs structure learning is inspired from classical methods of finding standard BNs structures as we have already discussed in Section 3.3. In Section 1.4, we have seen that BNs structure leaning algorithms are divided into three families, namely, constraint-based, score-based and hybrid approaches. Also we have deduced that hybrid approaches provides better experimental results, using several benchmarks and metrics. However, in the relational context, we find extensions of constraint-based and score-based approaches but no work has been proposed for relational hybrid approach. In this chapter we present our second contribution: an hybrid approach to learn the structure of a relational Bayesian network from a complete observational relational dataset. The proposal is an adaptation of the Max-Min Hill climbing (MMHC) algorithm (Tsamardinos et al., 2006) to the relational context. We call it Relational Max-Min Hill Climbing algorithm, RMMHC for short.

Contents

5.1 Introduction

RBNs extend standard BNs in the context of relational data and are suitable to deal with large-scale systems. On the other hand, MMHC demonstrated its performance when learning BNs from massive dataset. In (Tsamardinos et al., 2006), the authors have provided a wide comparative study with already existing approaches, either constraint-based or score-based, to learn BNs structure from observational data. Reported results have proved that MMHC outperforms other state-of-the-art approaches, using several benchmarks and metrics (execution time, SHD measure, etc.). In this chapter, we will present an extension of the MMHC algorithm to learn RBNs from relational data that we refer to as relational max-min-hill-climbing (RMMHC). The approach consists of a local search phase ensured by the relational max-min parents and children algorithm (RMMPC), and a global search phase ensured by the relational greedy search algorithm (Algorithm 14). Our learning assumptions stipulate that 1) the input contains a relational schema that describes the domain, 2) the training data consists of a fully specified instance of that schema. 3) The structure learning task comes to induce the dependency structure automatically from the complete relational observational training database.

The remainder of this chapter is as follows: Section 5.2 describes the local structure identification strategy. Section 5.3 presents the overall learning process. Section 5.4 gives the time complexity of the algorithms.

5.2 Relational Max Min Parents and children RMMPC

As seen in Section 1.4.4, the local search identification is composed of two main steps: the neighborhood identification realized by the $\overline{MMPC}$ algorithm, completed by a symmetrical correction. Our relational extension will preserve the same steps. In this section, we start by explaining the $\overline{RPPMC}$ algorithm for neighborhood identification. Then, we will detail the symmetrical correction for a relational domain. This latter is characterized by the non-symmetry caused by the use of aggregators. We present two different manners to construct the set of candidate parents and children (CPC) list of a target variable. We will show how this characteristic could be useful to automatically detect some directed parent or children relationships. Finally, we will discuss a conservative version of the algorithm.

5.2.1 Neighborhood identification: $\overline{RMMPC}$

At this step, we aim to find the list of neighbors of a target attribute T, that consists of either children or parents of T, from a set of potential variables.

When applying MMPC in the case of standard BNs, no difference is made between a node and a variable, and the potential set of parents and children of a node T is $\mathcal{V}\backslash T$, where $\mathcal{V}$ is the set of BN nodes. While, in a relational domain, and due to the horizon of crossed slot chains, the number of potential variables is not fixed. Note that we have to make the difference between an attribute and a variable:

- An attribute is characterized by its name, domain, a set of possible aggregators and the class that it belongs to.
- A variable is characterized by its name, domain, the class that it belongs to, a specific aggregator type and the slot chain that it is derived from.

Via Definition 3.2.3, a parent is a variable, while a child is an attribute. When searching the CPC(T) list of T, T is a target attribute, while $CPC(T)$ consists of the candidate parents and children of T. Each parent is a variable and each child is an attribute. $|CPC(T)|$ depends on the length of the traversed path $k \in \{0 \dots K_{max}\}$. For each value of k, a subset of potential parents and children can be generated. As the final generated $CPC(T)$ list may be very large, we adopt the same strategy as (Friedman et al., 1999a) and we proceed by phases. That is, suppose that we want to provide the list of children and parents of

Algorithm 22 $\overline{RMMPC}$

Require: $\mathcal{R}$: A relational schema, $\mathcal{I}$: A database instance
$Current\_slot\_chain\_length$: A slot chain length, T: A target attribute
Ensure: CPC: The set of parents and children of T, $CPC_T = CPC_T^{sym} \cup CPC_T^{asym}$
1: $Pot_{list} = Generate\_potential\_list(T, Current\_slot\_chain\_length)$
% Phase I: Forward
2: **repeat**
3: $\langle F.assocF \rangle = MaxMinHeuristic(T, CPC_T, Pot_{list})$
4: **if** $assocF \neq 0$ **then**
5: **if** $Current\_slot\_chain\_length = 0$ OR $does\_Not\_Contains\_Many\_Relationship(F)$ **then**
6: $CPC_T^{sym} = CPC_T^{sym} \cup F$
7: **else**
8: $CPC_T^{asym} = CPC_T^{asym} \cup F$
9: **end if**
10: $CPC_T = CPC_T^{sym} \cup CPC_T^{asym}$
11: $Pot_{list} = Pot_{list} \backslash F$
12: **end if**
13: **until** CPC has not changed or $assocF = 0$ or $Pot_{list} = \emptyset$
% Phase II: Backward
14: **for all** $A \in CPC_T$ **do**
15: **if** $\exists S \subseteq CPC, s.t. Ind(A; T|S)$ **then**
16: $CPC_T = CPC_T \backslash \{A\}$
17: **end if**
18: **end for**

each attribute T given a maximum slot chain length k_{max}, the neighborhood identification will be done on k phases where $k \in \{0 \dots k_{max}\}$. At phase 0, we will search for the set of parents and children of attribute T from the same class as T, at phase 1, we will search for the set of parents and children of attribute T in classes related to T class using reference slots. At phase 2, we will go through further classes and search for the set of parents and children of attribute T in classes related to T class using slot chains of length 2 and so on. The neighborhood identification, for one specified value of slot chain length, is described by Algorithm 22.

Potential list generation.

The $Generate\_potential\_list$ method (Algorithm 23) aims to identify the list of potential parents and children of a target attribute T given a slot chain length k. This method performs as follows.

- First, it searches all paths (i.e., slot chains) from a starting class X_T to other classes $Y \in \mathcal{X}$ of length k. These slot chains may be a sequence of reversed and not reversed slot references. As mentioned in (Getoor et al., 2007), if the slot chain contains at least one reversed slot reference then aggregators are needed. That is, the list of aggregators associated with each $A \in \mathcal{A}(Y)$ has to be used. The starting class is the class to which T, the target attribute, belongs.
- All attributes of reached classes Y, $A \in \mathcal{A}(Y)$ are considered as potential attributes. If $X = Y$, then all the attributes, expect T are considered as potential attributes. In fact, as (Friedman et al., 1999a), we only focus in generating dependency structures that are DAGs. Even if $\mathcal{S}$ may contain cycles, as discussed in Section 3.2.2, we do not consider this particular case where expert knowledge is needed to guarantee the acyclicity of derived ground Bayesian graph. As already mentioned, our input contains only a full specified relational observational database.
  The result of the $Generate\_potential\_list$ method is a set of potential variables defined as follows:
  - $X_T.A$, if $Y = X$ and we are searching for intra-class dependencies.
  - $\gamma(X_T.Slot\_Chain.Y.A)$, if the $Slot\_Chain\_length \neq 0$, where γ is an aggregation function. This

Algorithm 23 *Generate_potential_list*

Require: $\mathcal{R}$: A relational schema, $Current\_slot\_chain\_length$: A slot chain length, T: A target attribute
Ensure: Pot_{list}: the set of potential parents and children of T given $Current\_slot\_chain\_length$

```
Pot_list = ∅
ClassFromPath = FindPaths(X_T, R, Current_slot_chain_length)
for all ⟨Y ∈ X, SlotChain⟩ ∈ ClassFromPath do
    if Y ≠ X_T then
        listAttributes = findAllAttributes(Y)
    else
        listAttributes = findAllAttributes(Y)
        listAttributes = listAttributes\{T}
    end if
    for all A ∈ listAttributes do
        if SlotChain.length = 0 OR isAllNotReversed(SlotChain) then
            Pot_list = Pot_list ∪ X_T.SlotChain.Y.A
        else
            listAggregators = getAggregators(A)
            for all γ ∈ listAggregators do
                Pot_list = Pot_list ∪ γ(X_T.SlotChain.Y.A)
            end for
        end if
    end for
end for
```

set presents inter-class dependencies. With respect to Definition 3.2.3, $\gamma(X_T.Slot\_Chain.Y.A)$ could only be parents of the target attribute T.

Statistical tests. The asymmetry caused by the use of aggregators can provide more interesting interpretation than dependency detection. Clearly, if an independence is detected in both directions then we can conclude that these two variables are independent. Otherwise, if a dependence is detected in one direction and not in the other, the semantic of the slot chain involved in this dependency may even provide a decision on the dependency direction (i.e., identifying if the potential variable is either a potential parent or a potential child).

On the other hand, if we keep the same assumption as in the context of the $\overline{MMPC}$ algorithm, the symmetrical correction cannot be performed accurately: saying that $A \in CPC_T$ and verifying whether $T \in CPC_A$ is not yet enough in the relational context as the correlation is also related to the semantic of the slot chain. As some dependencies may require aggregators, there is an inherent asymmetry and this list of candidate dependencies is closely related to the slot chain composition and the perspective class.

To deal with these issues, we propose to divide the neighborhood list, CPC, into two sub-lists. Formally, $CPC(T) = CPC_T^{sym} \cup CPC_T^{asym}$, where:

- CPC_T^{sym} : The set of potential children and parents of target attribute T coming either from the same class as T, with slot chain length equal to 0 or from slot chains that do not contain any $Many$ relationship.
- CPC_T^{asym} : The set of potential variables coming from the other slot chains $\forall A \in CPC_T^2$, A could only be a potential parent of T as described in (Getoor, 2002).

As for the standard case, $MaxMinHeuristic$ (Algorithm 24) selects the variables that maximize the $MinAssoc$ with target attribute T conditioned to the subset of the currently estimated $CPC = CPC_T^{sym} \cup CPC_T^{asym}$.

Algorithm 24 The MaxMinHeuristic

Require: T: Target attribute, CPC: a subset of variables, Pot_{list}: The list of potential neighbors of T
Ensure: The variable F that maximizes the minimum association with T relative to CPC, and its association measurement $assocF$.
1: $assocF = max_{A \in Pot_{list}} MinAssoc(A; T|CPC)$
2: $F = argmax_{A \in Pot_{list}} MinAssoc(A; T|CPC)$

Algorithm 25 $RMMPC$

Require: $\mathcal{R}$: A relational schema, $\mathcal{I}$: A database instance, $Current\_slot\_chain\_length$: A slot chain length, T: A target attribute
Ensure: CPC: The set of parents and children of T, $CPC = CPC^{sym} \cup CPC^{asym}$
1: **if** $Current\_slot\_chain\_length = 0$ **then**
2: $CPC_T^{sym} = \emptyset, CPC_T^{asym} = \emptyset$
3: $CPC_T = CPC_T^{sym} \cup CPC_T^{asym}$
4: **end if**
5: $CPC_T = \overline{RMMPC}(\mathcal{R}, \mathcal{I}, T, Current\_slot\_chain\_length)$
6: **for all** $A \in CPC_T$ **do**
7: **if** $Current\_slot\_chain\_length = 0$ **then**
8: $CPC_A^{sym} = \emptyset, CPC_A^{asym} = \emptyset$
9: $CPC_A = CPC_A^{sym} \cup CPC_A^{asym}$
10: **end if**
11: $CPC_A = \overline{RMMPC}(\mathcal{R}, \mathcal{I}, A, Current\_slot\_chain\_length)$
12: **if** $A \in CPC_T^{sym}$ AND $T \notin CPC_A^{sym}$ **then**
13: $CPC_T = CPC_T \backslash \{A\}$
14: **end if**
15: **end for**

5.2.2 Symmetrical correction

RMMPC (Algorithm 25) comes to refine the result of Algorithm 22 by applying a symmetric verification to the result of $\overline{RMMPC}$. For standard BNs, this task returns to remove from each set $CPC(T)$ each node X for which $T \notin CPC(X)$. For RBNs, this task is quite different, as $CPC(T)$ consists of two subsets, the symmetrical correction depends on the concerned subset.

- For each $A \in CPC_T^{sym}$, we must verify that $T \in CPC_A^{sym}$, otherwise, A has to be removed from CPC_T^{sym}. This symmetrical correction is equivalent to the symmetrical correction of MMPC.
- For each $A \in CPC_T^{asym}$, we cannot apply the symmetrical correction since the SQL queries involved in such a case are not equivalent and the resulting datasets on which we will apply statistical tests are not the same. However, $\forall A \in CPC_T^{asym}$, A can only be a parent of T. By this way, we can deduce the dependency direction, directly from the first phase of RMMHC.

Example 5.2.1. *Let's consider the relational schema of figure 5.1 and the RBN of Figure 5.2 constructed among this relational schema.*

- *For CPC^{sym}, the symmetric correction is performed similar to the symmetric correction of MMPC (Tsamardinos et al., 2006).*
- *For the CPC^{asym}, the symmetric correction is not possible. In Figure 5.3, we are searching for $CPC_{ClassA.X_3}$ for a slot chain length $k = 1$. $\overline{RMMPC}_{ClassA.X_3} = \{\gamma(ClassA.ClassAPK^{-1}.ClassC.X_4)\}$, where γ is an aggregation function. On the other hand, if $ClassC.X_3 \in \overline{RMMPC}_{ClassA.X_4}$ then $\overline{RMMPC}_{ClassA.X_4} = \{ClassC.ClassAPK.ClassA.X_3\}$.*
  The statistical association computed for each retrieved dependency is not computed from the same dataset as the SQL requests used are not equivalent and consequently will produce two different

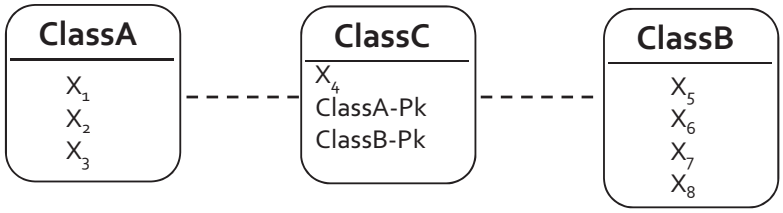


Figure 5.1: An example of a relational schema

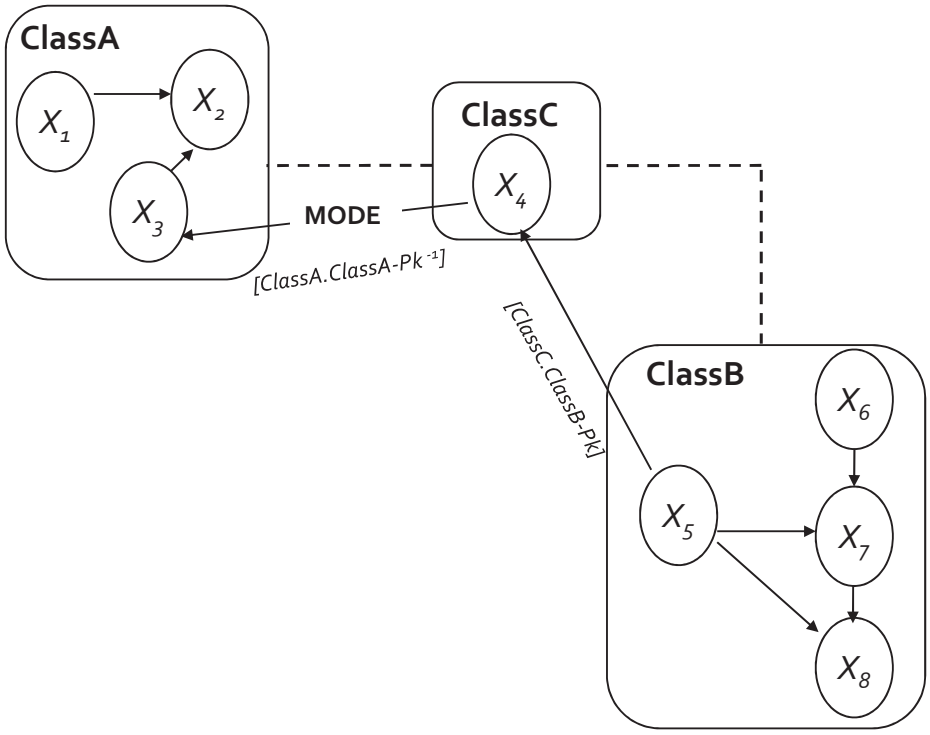


Figure 5.2: A RBN example of the relational schema of Figure 5.1

tables. In addition, with respect to Definition 3.2.3, $\gamma(ClassA.ClassAPK^{-1}.ClassC.X_4)$ could only be a potential parent of $ClassA.X_3$ and $ClassC.ClassAPK.ClassA.X_3$ could only be a parent of $ClassC.X_4$.

5.2.3 Conservative RMMPC

With small sample size, the power of statistical test may be limited and $RMMPC$ may lead to weak results. In Section 1.4.4, we have discussed a conservative approach that allows to deal with this issue. In Section 3.3.2, we have seen that the RCD algorithm compute statistical association in both directions and leaves the dependency if a statistical association is detected in at least one direction. In this section we propose the $RMMPC^c$, a conservative version of the $RMMPC$ algorithm that mixes both practices.

$RMMPC^c$ a slight variation of the $RMMPC$ algorithm that allows to provide stronger guarantees for true positives to enter the CPC list of T, especially when the algorithm is run with a limited sample size. The main differences between the $RMMPC$ and the $RMMPC^c$ lie in two levels:

– **At the $\overline{RMMPC}$ algorithm:** At this level we follow Maier et al. (Maier et al., 2013a) assumption, by computing the statistical association twice for asymmetric dependencies: A variable $\gamma(X_T.Slot\_Chain.Y.A)$ could enter the CPC list of a target attribute T if there is a statistical association $Assoc(\gamma(X_T.Slot\_Chain.Y.A), T|CPC(T))$ ***OR*** $Assoc(\gamma(Y_A.Slot\_Chain.X.T), A|CPC(T))$. The key difference is

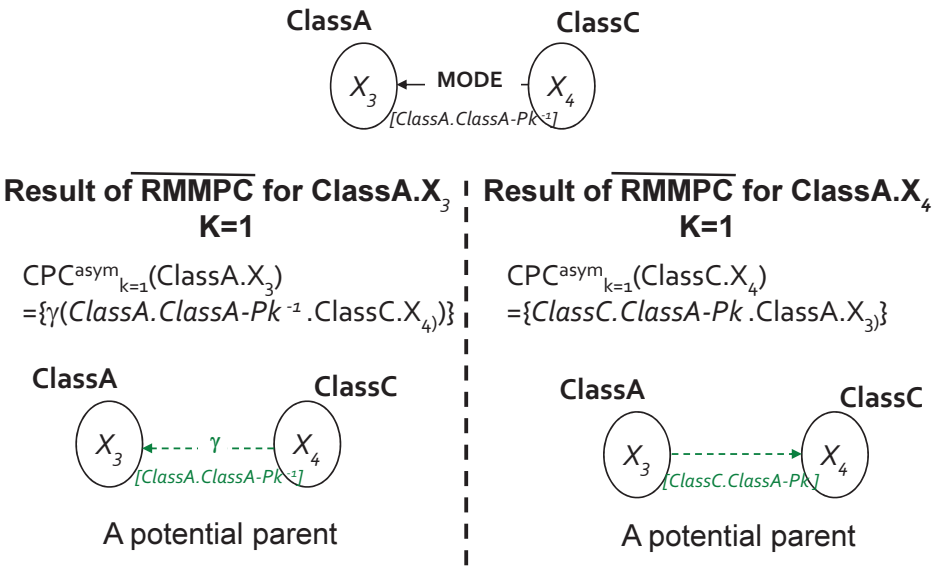


Figure 5.3: An example of asymmetric dependency

Algorithm 26 $RMMPC^c$

Require: $\mathcal{R}$: A relational schema, $\mathcal{I}$: A database instance, $Current\_slot\_chain\_length$: A slot chain length, T: A target attribute

Ensure: CPC: The set of parents and children of T, $CPC = CPC^{sym} \cup CPC^{asym}$

1: **if** $Current\_slot\_chain\_length = 0$ **then**
2: $CPC_T^{sym} = \emptyset, CPC_T^{asym} = \emptyset$
3: $CPC_T = CPC_T^{sym} \cup CPC_T^{asym}$
4: **end if**
5: $CPC_T = \overline{RMMPC}^c(\mathcal{R}, \mathcal{I}, T, Current\_slot\_chain\_length)$
6: $Pot_{list} = Generate\_potential\_list(T, Current\_slot\_chain\_length)$
7: **for all** $A \in Pot_{list} \backslash CPC_T$ **do**
8: **if** $Current\_slot\_chain\_length = 0$ **then**
9: $CPC_A^{sym} = \emptyset, CPC_A^{asym} = \emptyset$
10: $CPC_A = CPC_A^{sym} \cup CPC_A^{asym}$
11: **end if**
12: $CPC_A = \overline{RMMPC}^c(\mathcal{R}, \mathcal{I}, A, Current\_slot\_chain\_length)$
13: **if** $T \in CPC_A^{sym}$ **then**
14: $CPC_T^{sym} = CPC_T^{sym} \cup \{A\}$
15: **end if**
16: **end for**

then in the MaxMinHeuristic treatment. We refer to this conservative version as $MaxMinHeuristic^c$. $\overline{RMMPC}^c$ is $\overline{RMMPC}$ that calls $MaxMinHeuristic^c$.

– **At the symmetrical correction:** At this level we follow (de Morais and Aussem, 2010) assumption, by applying an OR condition, instead of the AND condition when performing symmetrical correction for the CPC^{sym} subset. A variable $A \in CPC_T^{sym}$ if $A \in CPC_T^{sym}$ OR $T \in CPC_A^{sym}$, which means that, after performing the symmetric correction, $CPC_T^{sym} = CPC_T^{sym} \cup \{\forall A | T \in CPC_A^{sym}\}$.

The $RMMPC^c$ algorithm is depicted by algorithm 26.

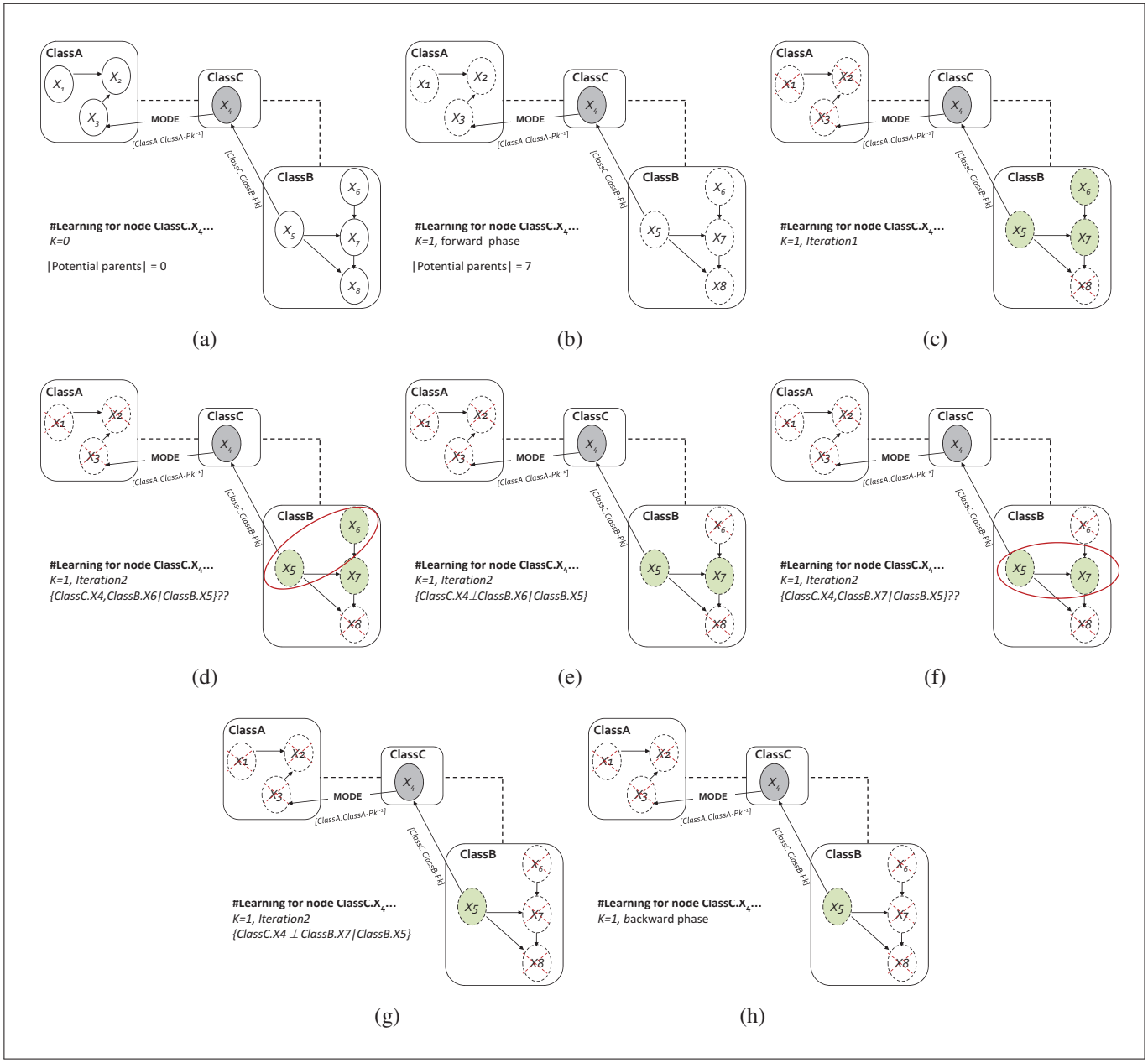


Figure 5.4: Example trace of $\overline{RMMPC}$ with target node $ClassC.X_4$

5.2.4 Toy example

In this section, we provide an example trace of RMMPC with a toy example, the relational schema as well as the RBN used are respectively depicted by Figures 5.1and 5.2.

The relational observational dataset $\mathcal{I}$ is sampled from the network of Figure 5.2, with an average of 1000 records per table. The *Assoc function* used for independence measurement is the *Mutual Information*. To simplify, we suppose that the $MODE$ is the only possible aggregator for all attributes. The longest slot chain present in the model is of length 1, we fix the maximum slot chain length to $K_{max} = 1$. The algorithm performs on two phases, $k = 0$ and $k = 1$.

Figure 5.4 provides an example trace of $\overline{RMMPC}$ with target node $ClassC.X_4$.

– Initially, $CPC(ClassC.X_4) = \emptyset$ and it remains empty for $k = 0$ (Figure 7.6(a)) as $ClassC.X_4$ is the only attribute of $ClassC$.

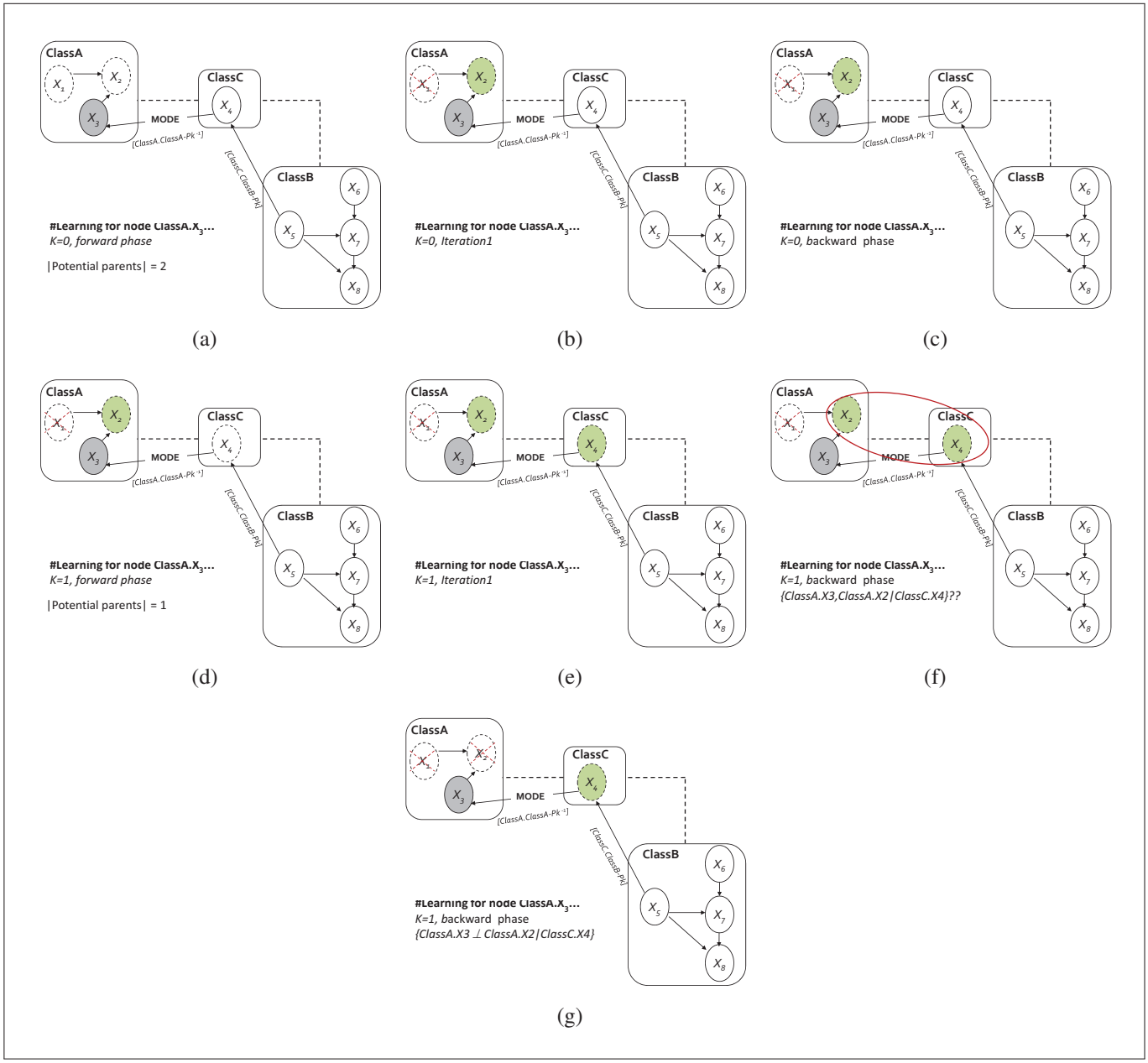


Figure 5.5: Example trace of $\overline{RMMPC}$ with target node $ClassA.X_3$

– For $k = 1$, $ClassC.X_4$ has 7 possible parents (dotted nodes of Figure 7.6(b)), namely, $Pot_{list}(ClassC.X_4) = \{$***ClassC.ClassA-PK***$.ClassA.X_i, \forall i \in \{1,\ldots,3\}\} \cup \{$***ClassC.ClassB-PK***$.ClassB.X_j, \forall j \in \{5,\ldots,8\}\}$. At the first iteration of the forward phase $CPC(ClassC.X_4) = \{$***ClassC.ClassB-PK***$.ClassB.X_i, \forall i \in \{5,\ldots,7\}\}$ (Figure 7.6(c)). At the second iteration, computing the statistical association between $ClassC.X_4$ and ***ClassC.ClassB-PK***$.ClassB.X_6$ conditionally to ***ClassC.ClassB-PK***$.ClassB.X_5$ (Figure 7.3(a)) leads to eliminate ***ClassC.ClassB-PK*** $.ClassB.X_6$ from $CPC(ClassC.X_4)$ (Figure 7.3(b)). Also, computing the statistical association between $ClassC.X_4$ and ***ClassC.ClassB-PK***. $ClassB.X_7$ conditionally to ***ClassC.ClassB-PK***$.ClassB.X_5$ (Figure 7.3(c)) leads to eliminate ***ClassC. ClassB-PK*** $.ClassB.X_7$ from $CPC(ClassC.X_4)$ (Figure 5.4(g)).
– The result of $\overline{RMMPC}$ algorithm for node $ClassC.X_4$ is $CPC(ClassC.X_4) = \{$***ClassC.ClassB-PK*** $.ClassB.X_5\}$ (Figure 5.4(h)).

Figure 5.5 provides an example trace of $\overline{RMMPC}$ with target node $ClassA.X_3$.

– For $k = 0$, $ClassA.X_3$ has 2 possible parents (dotted nodes of Figure 5.5(a)), namely, $Pot_{list}(ClassA.X_3) = \{ClassA.X_1, ClassA.X_2\}$. At the end of the first phase (i.e., $k = 0$), $CPC(ClassA.X_3) =$

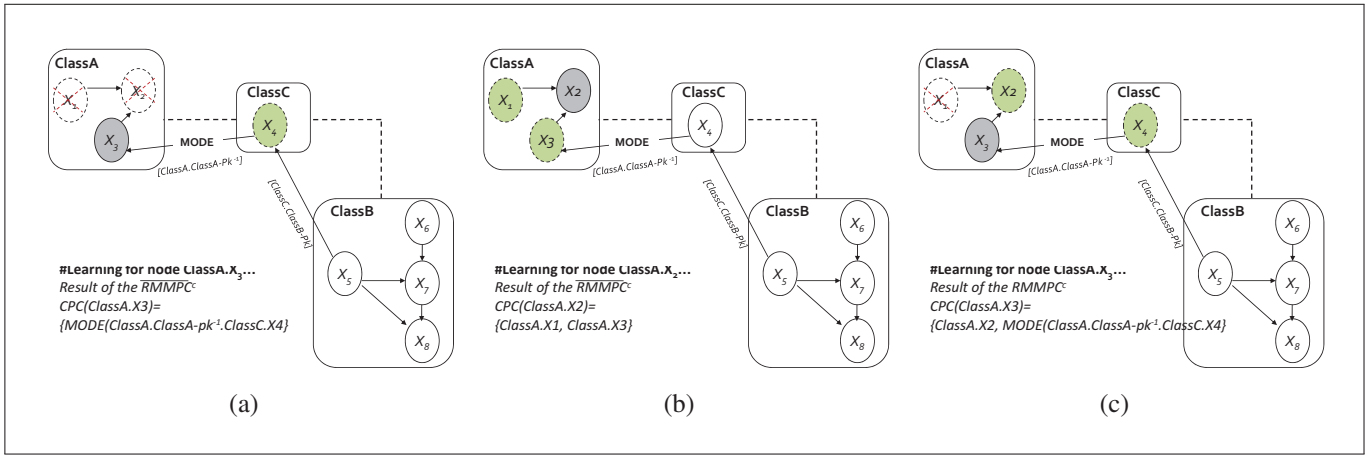


Figure 5.6: Example trace of $RMMPC^c$ with target node $ClassA.X_3$

$\{ClassA.X_2\}$ (Figure 5.5(c)).

- For $k = 1$, $Pot_{list}(ClassA.X_3) = \{MODE(\textbf{\textit{ClassA.ClassA-PK}}^{-1}.ClassC.X_4)\}$ (dotted node of Figure 5.5(d)). At the forward phase of the first iteration $CPC(ClassA.X_3) = \{ClassA.X_2, MODE(\textbf{\textit{ClassA.ClassA-PK}}^{-1}.ClassC.X_4)\}$ (Figure 5.5(e)). At the backward phase, computing the statistical association between $ClassA.X_3$ and $ClassA.X_2$ conditionally to $MODE(\textbf{\textit{ClassA.ClassA-PK}}^{-1}.ClassC.X_4)$ (Figure 5.5(f)) leads to eliminate $ClassA.X_2$ from $CPC(ClassA.X_3)$ (Figure 5.5(g)).
- The final $CPC(ClassA.X_3) = \{MODE(\textbf{\textit{ClassA.ClassA-PK}}^{-1}.ClassC.X_4)\}$. $ClassA.X_2$ should had remained in $CPC(ClassA.X_3)$, however the test of independence conditionally to $MODE(\textbf{\textit{ClassA.ClassA-PK}}^{-1}.ClassC.X_4)$ has removed it. This may be due to to the limited sample size. Even if the result of the $\overline{RMMPC}$ algorithm applied to $ClassA.X_2$ gives that $ClassA.X_3 \in CPC(ClassA.X_2)$, it will be removed at the symmetrical correction step of the $RMMPC$ algorithm.

We have run the $RMMPC^c$ algorithm (cf. Algorithm 26) using the same dataset. Figure 5.6 provides an example trace of $RMMPC^c$ with target node $ClassA.X_3$. The $\overline{RMMPC}^c$ result is $CPC(ClassA.X_3) = \{MODE(\textbf{\textit{ClassA.ClassA-PK}}^{-1}.ClassC.X_4)\}$ (Figure 5.6(a)). At the symmetrical correction step, the $RMMPC^c$ will verify whether $ClassA.X_3 \in CPCX|X \notin CPC(ClassA.X_3)$. As $ClassA.X_3 \in CPC(ClassA.X_2)$ (Figure 5.6(b)), $ClassA.X_2$ will enter $CPC(ClassA.X_3)$.

5.3 Relational Max Min Hill-Climbing: RMMHC

5.3.1 Global structure identification

The global structure identification is performed using a score-based algorithm only on the set of variables derived from the first local search phase. We choose to work with the relational greedy search RGS procedure as described by Algorithm 14, using the relational Bayesian score described in Section 3.3.1.

In this case, $Pot_K(X.T)$ consists of the CPC list of variable $X.T$ found on the local search step. As this set contains two subsets, the choose of the operator to be performed during the neighbors generation process will depends on the concerned subset:

- For CPC_T^{sym}: each $A \in CPC_T^{sym}$ can be either a child or a parent of $X.T$ so all the operators, namely, add_edge, $delete\_edge$ and $reverse\_edge$ can be tested.
- For CPC_T^{asym}: each $A \in CPC_T^{asym}$ is a potential parent of $X.T$ so only the add_edge and $delete\_edge$ operators can be tested.

Algorithm 27 $RMMHC\_single\_GS$

Require: $\mathcal{R}$: A relational schema, $\mathcal{I}$: A database instance, k_{max}: $Maximun\_SlotChain\_Length$
Ensure: The local optimal dependency graph $\mathcal{S}$
% Local search
1: **for** $Current\_slot\_chain\_length = 0$ to k_{max} **do**
2: **for all** T **do**
3: $CPC_T = RMMPC(\mathcal{R}, \mathcal{I}, T, Current\_slot\_chain\_length)$
4: **end for**
5: **end for**
% Global search
6: Perform Relational Greedy Search with operators add-edge, delete-edge, reverse-edge.
Only try operators add-edge and delete-edge $A \rightarrow T$ if $A \in CPC_T^{asym}$.

5.3.2 The overall algorithm

The global search step is an expensive step in term of complexity, since the size of the generated neighborhood may increase rapidly. In the standard MMHC, the MMPC result is used as input for the GS algorithm. In the relational context both the local and global search procedures are included in an iterative process which increases the slot chain length where we look for possible probabilistic dependencies between variables and till we reach a maximum value of slot chain length which is user-defined. So, two patterns can be envisaged:

- Iterate on the slot chain length during only the local search procedure then perform only one global search using the results of the iterative local search.
- Perform the local search as well as the global one for each slot chain length.

The RMMHC algorithm can be performed following one of these two possible assumptions:

- **With single global search call.** In this case the local search procedure is performed in phases until reaching the maximum slot chain length. The result of this search procedure will be the CPC list of all variables with different possible slot chains lengths. This result is the input of the global search procedure that will be run only one time. The overall process is as presented in Algorithm 27.
- **With multiple global search calls.** In this case both the local and global search procedures will be performed in phases until reaching the maximum slot chain length. At each phase K, a CPC list is identified with slot chain length equal to K and passed to the global search procedure. This latter starts, at phase $K = 0$, from the empty graph. Then, at each new phase K, the global search is performed using the graph structure found at phase $k - 1$. The overall process is as presented in Algorithm 28.

The first assumption seems to be more suitable with MMHC principle as for this latter the global search is performed once, on all candidate nodes derived from the local search step. This allows to push the global search phase to the end of the process, nevertheless, this does not guarantee the use of the correct skeleton when identifying the CPC lists.

The second assumption is more suitable with the relational context as it allows to treat each slot chain length separately during the learning process. This approach allows to identify and clean the neighborhood at each iteration, however, the greedy search is called multiple times.

We have to mention that for both versions of the RMMHC algorithm, we apply the relational greedy search algorithm (see Algorithm 14) while omitting the outer loop, as the slot chain concern will be taken into consideration by the $RMMHC$ algorithm.

Algorithm 28 $RMMHC\_Multiple\_GS$

Require: $\mathcal{R}$: A relational schema, $\mathcal{I}$: A database instance, k_{max}: $Maximun\_SlotChain\_Length$
Ensure: The local optimal dependency graph $\mathcal{S}$

```
for Current_SlotChain_Length = 0 To k_max do
    % Local search
    for all T do
        CPC_T = RMMPC(R, I, T, Current_slot_chain_length)
    end for
    % Global search
    Perform Relational Greedy Search with operators add-edge, delete-edge, reverse-edge.
    Only try operators add-edge and delete-edge A → T if A ∈ CPC_T^{asym}.
end for
```

5.4 Time complexity of the algorithms

The MMPC (cf. Section 1.4.4) consists of the $\overline{MMPC}$ algorithm of complexity $\mathcal{O}(|Pot_{list}|\,.2^{|CPC|})$ and an additional symmetrical correction. Thus, its overall complexity is $\mathcal{O}(|Pot_{list}|^2\,.2^{|CPC|})$.

At each iteration of the classical greedy search algorithm (cf. Section1.4.3), the number of possible local changes is bounded by $\mathcal{O}(\mathcal{V}^2)$, where $\mathcal{V}$ is the number of variables in the graph.

Our $\overline{RMMPC}$ algorithm presents the same steps as for the standard case, augmented with the $Generate\_potential\_list$ procedure which is of complexity $\mathcal{O}(N^k)$, where N is the number of classes and k is the current slot chain length. Thus, its time complexity, at each k value, $k \in \{0 \dots K_{max}\}$ remains equal to $\mathcal{O}(|Pot_{list}|\,.2^{|CPC|})$. Thus, augmented with the symmetrical correction, the time complexity of the RMMPC algorithm is $\mathcal{O}(|Pot_{list}|^2\,.2^{|CPC|})$.

For RGS, we have to iterate on variables and for each variable, we have to iterate on the list of all potential parents of this variable. Unlike the standard case where this set is fixed for all nodes of the graph, the relational context makes this set variable, depending on the length of the considered slot chain. A probabilistic dependence is no longer a simple link between two nodes in a graph, but a link from a possible chain slot value. Let us consider β the number of potential parents that could be reached, then the number of possible local changes is bounded by $\mathcal{O}(\beta.\mathcal{V})$. Note that $\beta = |CPC|$ when the RGS is called after a local search step performed using $RMMPC$ algorithm. We distinguish between $\beta_{max} = |CPC|$ for RMMHC algorithm with single global search and $\beta_k = |CPC|$ for $RMMHC$ algorithm with multiple global search, $\forall k \in \{0 \dots K_{max}\}$.

In RMMHC algorithm, either with single or multiple global search, the local search step has been augmented with an outer loop presenting the current slot chain length to consider at each iteration. Thus the final complexity of the local search is $\mathcal{O}(K_{max}.\,|Pot_{list}|^2\,.2^{|CPC|})$. It is obvious that the RMMHC^c algorithm preserves the same complexity as the RMMHC algorithm.

5.5 RMMHC vs Related work

In this section, we highlight assumptions made either by RMMHC or by related state-of-the-art methods and discuss some perspectives. Table 5.1 presents the main particularity of each approach.

Causal sufficiency. RGS, RCD and RMMHC make the causal sufficiency assumption which states that for all considered relational models, all common causes of observed variables are also observed and included in the model.

Addressing self-relationships. Self-relationships are defined by Maier et al. (Maier et al., 2013c) as

Algorithm	RGS	RCD	RMMHC	RMMHC^c
Family	score-based	constraint-based	hybrid	hybrid
Input	relational schema	relational schema with ER description	relational schema	relational schema
Output	learned RBN	list of canonical dependencies	learned RBN	learned RBN
Testing dependency	using score functions	using statistical tests	using score functions and statistical tests	using score functions and statistical tests
Statistical association measurement	RBD score	Linear regression	RBD score, Linear regression, Mutual information	RBD score, Linear regression, Mutual information
Heuristic	AND condition	OR condition	AND condition	OR condition
How to do?	given a k_{max}, it generates potential dependencies iteratively, for each $k \in \{0 \dots k_{max}\}$	given a k_{max}, it generates all the potential dependencies at once	given a k_{max}, it generates potential dependencies iteratively, for each $k \in \{0 \dots k_{max}\}$	given a k_{max}, it generates potential dependencies iteratively, for each $k \in \{0 \dots k_{max}\}$
Self-relationship	yes	no	yes	yes
Structural uncertainty	no	no	no	no
cyclic models	no	no	no	no

Table 5.1: Overview of RGS, RCD, RMMHC and RMMHC^c particularities

relationship classes that involve the same entity class more than once. Even if the authors have defined relational schemas that can express these types of relationships. for simplicity, they do not consider them when searching for the list of potential dependencies of the RCD algorithm (cf. Section 3.3.2). RCD is executed around the AGG specification which is constructed using the extend method (cf. Definition 3.2.23). The extend method relies on the definition of relational paths (cf. Definition 3.2.13) which requires unique entity class names within $[E, R, E]$ triples. However, in a relational context, self-relationships are frequently involved (e.g., scholarly articles cite other articles). In contrast to RCD, both RGS and RMMHC support this additional layer of complexity.

Cyclic models. For the three algorithms that we discuss, the relational model is assumed to be acyclic with respect to the class dependency graph (cf. Definition 3.2.7). Note that, as discussed in (Friedman et al., 1999a), this additional layer of complexity can be taken into account for further improvement of these algorithms by studying stratified dependency structures (cf. Definition 3.2.9), where prior knowledge allowing us to guarantee the legality of certain dependency models is needed.

Structural uncertainty. RGS, RCD and RMMHC algorithms do not treat structural uncertainty (cf. Section 3.2.6). They only search for probabilistic dependencies between attributes. In (Maier et al., 2010), existence uncertainty has been addressed by the RPC algorithm. Yet, the proposed algorithm was not sound and complete. RCD comes to refine RPC but it omit the existence uncertainty issue and leave it for future research.

5.6 Conclusion

This chapter proposed a new hybrid approach to learn RBN structures. We have proposed two possible versions of the algorithm and we have discussed its complexity. We have also made a discussion with respect to related work.

In the next chapter, we will move on describing the implementation process. We will introduce our software project and provide some details on implemented statistical tests and score functions.

6

Implementation

The contributions detailed in chapters 4 and 5 have to be approved experimentally. However, before proceeding the experiments, and because of the great efforts that have been devoted to the development, we dedicate this chapter to introduce our PILGRIM software under which we have implemented all the proposed contributions.

Contents

6.1 Introduction

This chapter is dedicated to the implementation of our theoretical contributions. As seen in Section 3.5, even if there is some software tools that support relational data representation, these tools either do not support relational structure or provide it for RBN similar models for which the model specification is not necessary similar to RBN. Furthermore, these softwares do not provide relational evaluation metrics. Thus, in order to evaluate our contributions, we implemented all the proposed algorithms into the C++ PILGRIM project which is under development. The PILGRIM software tool allows to deal with several probabilistic graphical models. It presents the development effort of several participants who have made or are making their academic researchers within the Data User KnowledgE (DUKe) research group of the LINA lab [1]. Section 6.1 provides an overview of the PILGRIM project and lists the used environment and softwares. Then, Section 6.3 focuses especially in the PILGRIM Relational framework and illustrates different functionalities that have already been implemented.

6.2 The PILGRIM project

PILGRIM is actually under development to provide an efficient tool to deal with several probabilistic graphical models (e.g., BNs, RBNs). In this section we give a brief representation of the PILGRIM project and we specify additional libraries used by PILGRIM.

6.2.1 PILGRIM in nutshell

PILGRIM is a software tool for modeling, learning and reasoning upon probabilistic networks. It has support to:

1. **Probabilistic networks modeling:** We are particularly interested with directed acyclic graphs. Actually, PILGRIM allows to model:
   - Bayesian Network (BN)
   - Relational Bayesian Network (RBN)
   - Relational Bayesian Network with Reference Uncertainty (RBN-RU)
2. **Probabilistic networks learning:** Besides models representation, Pilgrim addresses the main learning tasks related to those models, namely:
   - Parameter learning
   - structure learning
3. **Probabilistic networks benchmarking:** The validation of any learning approach goes through an evaluation process where benchmarks availability is essential. Thus, PILGRIM allows:
   - Model generation
   - Sampling
4. **Probabilistic networks evaluation metrics:** There are several metrics allowing to evaluate probabilistic graphical models structure learning algorithms. PILGRIM implements a set of metrics to evaluate the quality of reconstruction of the graph structure, namely:
   - Distance-based measures
   - Performance measures

PILGRIM is developed around four main sub-projects, namely PILGRIM general, PILGRIM structure learning, PILGRIM relational and PILGRIM applications such illustrated by Figure 6.1. The PILGRIM general was the first developed project it allows modeling standard probabilistic models.

1. pilgrim.univ-nantes.fr

Project managers: Philippe LERAY and Thomas VINCENT					
Sub-project	**Code lines number**	**Developers**	**Functionalities**	**Role**	
				Leader	Participant
PILGRIM general	6196	Amanullah YASIN (1) Ghada TRABELSI (2)	BN specification	(1)	(2)
PILGRIM structure learning	6175	Amanullah YASIN (1) Ghada TRABELSI (2)	BN structure learning	(1)	(2)
PILGRIM relational	27412	Anthony COUTANT (1) Mouna BEN ISHAK (2) Rajani CHULYADYO (3)	RBN specification	(1)	(2), (3)
			RBN-RU specification	(1)	
			RBN parameters learning learning	(2)	(1)
			RBN serialization	(1)	(2)
			RBN visualization	(1)	
			RBN unserialization	(2)	
			RBN structure learning	(2)	
			RBN structure learning evaluation metrics	(2)	
			RBN benchmark random generation	(2)	
PILGRIM applications	1563	Rajani CHULYADYO (1)	RBN-based recommender system	(1)	

Table 6.1: PILGRIM Overview: projects, code lines number, functionalities & main contributor roles

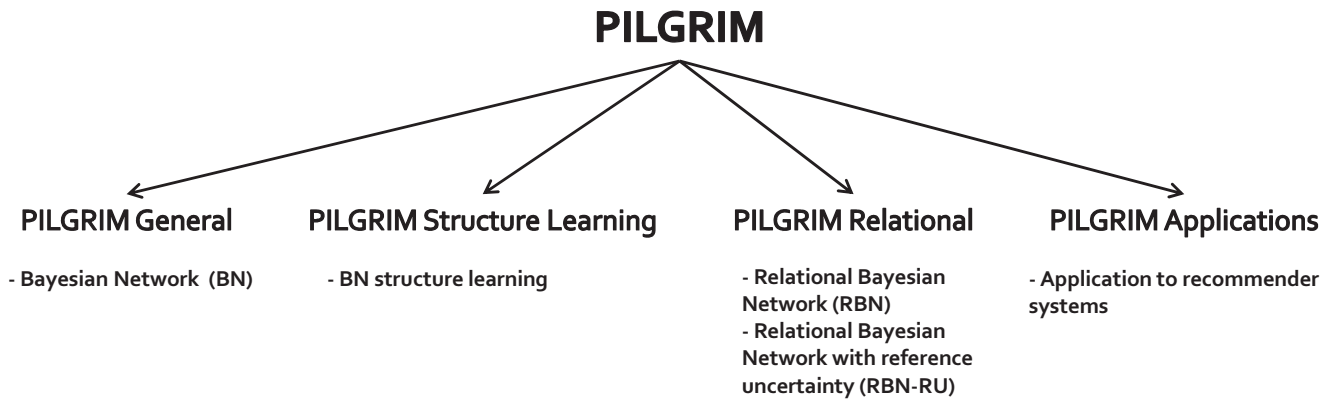


Figure 6.1: Overview of the PILGRIM project

By standard probabilistic models we refer to the family of probabilistic networks which are constructed among the classical propositional data representation.The PILGRIM structure learning project provides several structure learning algorithms for BNs (e.g., GS, MMHC). It also implements several score functions (e.g., BDeu, BIC, AIC, MDL), statistical tests (e.g., mutual information) and metrics to evaluate the quality of the reconstructed networks (e.g., SHD). The PILGRIM relational project provides several functionalities to deal with probabilistic networks for relational data representation. Finally, PILGRIM application contains some domain applications (e.g., RBN application to recommender systems) where probabilistic networks implemented by either PILGRIM general or PILGRIM relational have been used. Table 6.1 recapitulates the functionalities developed under PILGRIM while presenting the main participants and their roles.

6.2.2 Additional libraries

The PILGRIM project uses several existing C++ libraries to manage graph structures, to manipulate BNs objects, to communicate with a RDBMS or to run unit tests.

The Boost Graph Library (BGL). BGL is a C++ open source library that provides a generic open interface for traversing graphs. Its implementation follows a generic programming principle and its source can be found as part of the Boost distribution [2]. BGL is characterized by its easy of use and integration in any program: no need to be built to be used, wealth of documentation and multiple code examples. It consists of a set of core algorithm patterns, namely, Breadth First Search, Depth First Search and Uniform Cost Search, and a set set of graph algorithms (e.g., Dijkstra's Shortest Paths, Connected Components, Topological Sort).

Database Template Library (dtl). dtl is a C++ open source library. The specificity of this library is that it can run on multiple platforms and C++ compilers. In addition, Several DBMS are supported (Oracle, PostgreSQL, MySql, etc) and Database access is ensured through ODBC drivers. dtl allows to perform several manipulations on databases such as reading and writing records. It also allows to perform more other requests such as creating schemas, tables, constraints, etc. dtl is well documented and a variety of examples are given and commented. Moreover, instructions for using the library are provided and precision on how to use it with each DBMS is given [3].

Googletest. Released under the BSD 3-clause license [4], Google Test presents a library for writing C++ unit tests. It works on a variety of platforms and can be easily integrated to any c++ program. The library allows several test types and several options for running the tests [5].

2. http://sourceforge.net/projects/boost/files/
3. http://dtemplatelib.sourceforge.net/
4. http://opensource.org/licenses/BSD-3-Clause
5. https://code.google.com/p/googletest/

Besides, when we instantiate RBNs, ground Bayesian networks are BN objects defined using the ProBT library 1.6.

6.2.3 Data accessibility

Some functionalities (e.g., parameter learning, structure learning) involve datasets as input. For standard probabilistic models, flat data representation (e.g., a text file) is used. For relational probabilistic models, relational data representation is needed. For the second case, we use the PostgreSQL Relational database management system.

PostgreSQL. Postgres is an open-source object-relational database management system. Initially created at the University of California at Berkeley, PostgreSQL is now considered among the most advanced open-source database. It supports a large part of the SQL standard and provides the possibility to be used, modified, and distributed by anyone free of charge for any purpose, be it private, commercial, or academic [6].

The next section is dedicated to the PILGRIM relational project. We will briefly present the main modules of PILGRIM relational. Then, we will focus on the main modules on which we have contributed.

6.3 PILGRIM Relational modules

PILGRIM relational is built on the RBN specification, thus it requires the data to be modeled through a relational schema. The probabilistic structure defines the dependencies among the probabilistic attributes of that schema. Reference slots encode One-to-Many cardinality type, therefore it is possible that some children nodes have multiple parents for the same probabilistic dependency. As multiple parents have to be aggregated, PILGRIM relational defines several aggregation functions and Multi-set operators (e.g., AVERAGE, MODE, INTERSECTION) (cf. Section 2.3.1). PILGRIM relational allows also to model and reason about RBN-RU (cf. Section 3.2.6). Figure 6.2 presents the main modules of the PILGRIM relational project. In the remaining of this section we will especially focus on our contribution on this project.

Under PILGRIM Relational, we have already developed tools to define, instantiate and visualize RBNs. These latter may be defined either manually or from a relational database schema or from an existing RBN object serialized as an XML file. We use existing methods from the PILGRIM general project to perform probabilistic inference at the ground Bayesian network level. We have also developed parameter learning as well as structure learning approaches for RBNs from complete relational observational datasets. We have also implemented modules for saving and reconstructing RBN objects. The PILGRIM relational contains the implementation of our generator tool as well as our RBN structure learning evaluations measurements.

6.3.1 RBN serialization and unserialization

Methods to save a RBN object into a stored representation (e.g., file, memory buffer) or to create RBN object from stored representation are very important and useful. Therefor, we have implemented a serialization method which intends to save RBN objects into XML files and an unserializaton method which intends to reconstruct them later either in the same computer environment or in another one. In order to facilitate the reusability of our objects, we have worked with an enhanced version of an already existing XML format for probabilistic graphical models encoding, namely the ProbModelXML format. Here we start by presenting the main properties of the chosen format. Then, we present extensions made to encode RBNs.

6. http://www.postgresql.org/docs/9.4/interactive/index.html

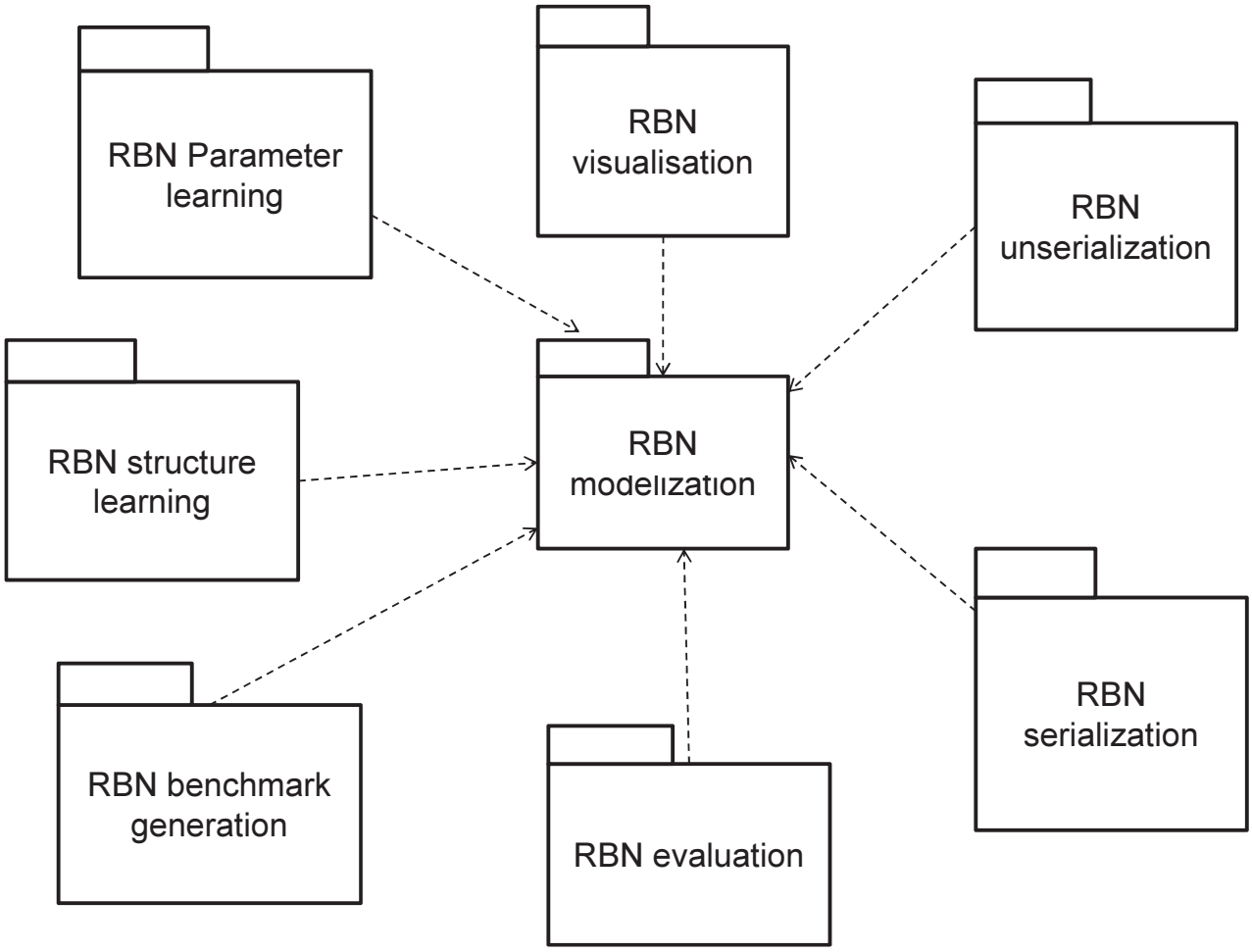


Figure 6.2: Package diagram of the PILGRIM Relational project

```
<ProbNet type=enumNetworkType >
   <AdditionalConstraints />0..1
   <Comment />0..1
   <DecisionCriteria />0..1
   <Agents />0..1
   <Language />0..1
   <AdditionalProperties />0..1
   <Variables />
   <Links />
   <Potentials />
</ProbNet>
```

Figure 6.3: The skeleton for a probabilistic network encoded using the ProbModelXML format

6.3.1.1 The XML encoding of serialized RBNs

There is currently no formalization of RBNs, so that our generated models are serialized based on the *ProbModelXML* format for encoding probabilistic graphical models. We use an enhanced version of the XML syntax of the *ProbModelXML* specification (Arias et al., 2011).

```
<ProbNet type="rbn-RBN">
  + <rbn-Classes>
  + <rbn-ClassesLinks>
  + <Variables>
  + <Links>
  + <Potentials>
</ProbNet>
```

Figure 6.4: The skeleton for a RBN using the ProbModelXML format

Probabilistic networks encoding in the ProbModelXML format ProModelXML is a prespecified XML format allowing to encode several probabilistic graphical models (e.g., Bayesian networks, Markov networks, Dynamic Bayesian networks). It has been developed by the Research Center on Intelligent Decision-Support Systems of the UNED in Madrid in 2011. The format is supported by *OpenMarkov* software tool [7]. The ultimate goal behind ProbModelXML is to provide a common format for the interchange of PGMs. Therefore, it gives the possibility of encoding other probabilistic graphical models without a necessity to modify the format definition. The skeleton for a probabilistic network is as depicted by Figure 6.3.

RBNs encoding using the ProbModelXML format The XML syntax that we used to encode RBNs under PILGRIM is an enhanced version of the ProbModelXML specification. Mainly, we have added new tags to encode properties related to the meta-model specification (i.e., classes and reference slots) and we have used the $\langle AdditionalProperties \rangle$ tags to add further notions related to RBNs (i.e., aggregators and slot chains associated with dependencies, classes associated with nodes). The skeleton for a relational probabilistic network is as depicted by Figure 6.4.

The additional network properties concerns classes and reference slots representation. An example is provided by Figure refrbnXML1.

- **Classes:** The $\langle rbn - Classes \rangle$ tag allows to encode classes of the rbn. A class has as attribute $name$, a string value.
- **Links between classes:** The $\langle rbn - ClassesLinks \rangle$ tag allows to encode reference slots. A link has three attributes fk, to and $from$, all of type string. The fk attribute specifies the reference slot name. The to attribute specifies the name of the referenced class. The $from$ attribute specifies the name of the referencing class.

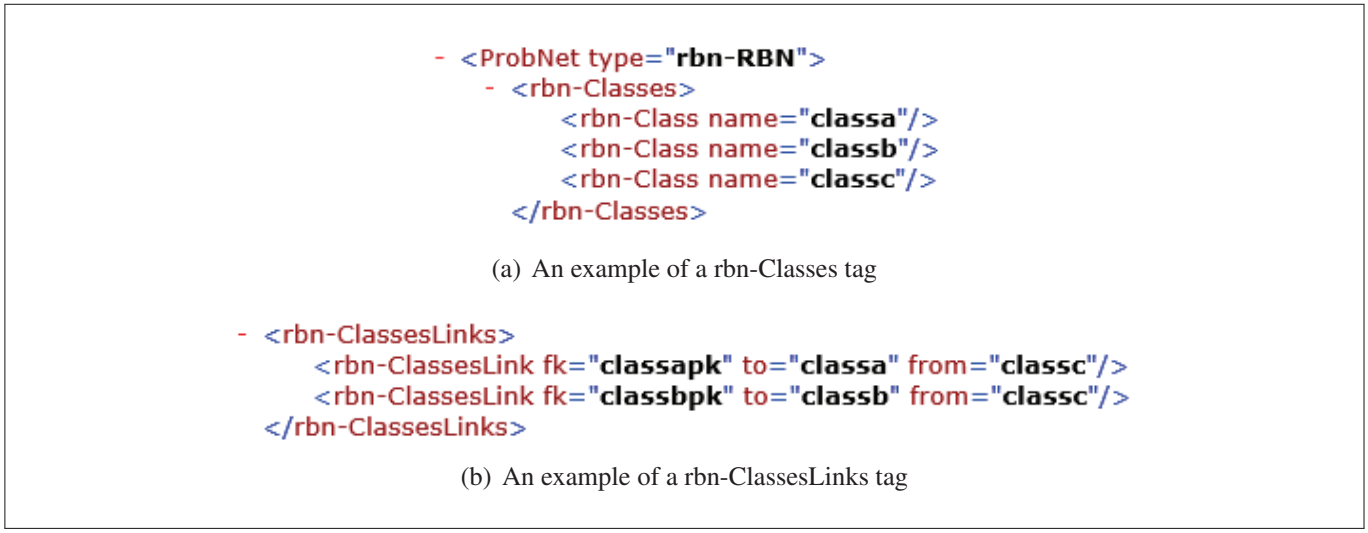
The already existing properties on which we have made some modifications are variable, link and potential. An example is provided by Figure 6.6.

- **Variable:** The $\langle AdditionalProperties \rangle$ tag of a variable allows to associate a class with a node.
- **Link:** The $\langle AdditionalProperties \rangle$ tag of a link allows to associate a slot chain and an aggregator, if any, with a dependency.
- **Potential:** To specify that the potential comes from an aggregated dependency, a new attribute $rbn - aggregator$ has been defined within the $\langle Variable \rangle$ tag of a potential.

6.3.2 Parameter learning

Given a dependency structure $\mathcal{S}$, parameter learning comes to estimate the CPDs of this structure. We have developed parameter learning approaches for RBNs from complete relational observational dataset. Despite its straightforwardness, it is an interesting task as it represents a crucial component to define score functions used by structure learning algorithms. RBNs parameters learning approaches are inspired from statistical and Bayesian parameter learning methods of classical Bayesian networks. The statistical approach consists in estimating the probability of an event by its frequency of occurrence in the database,

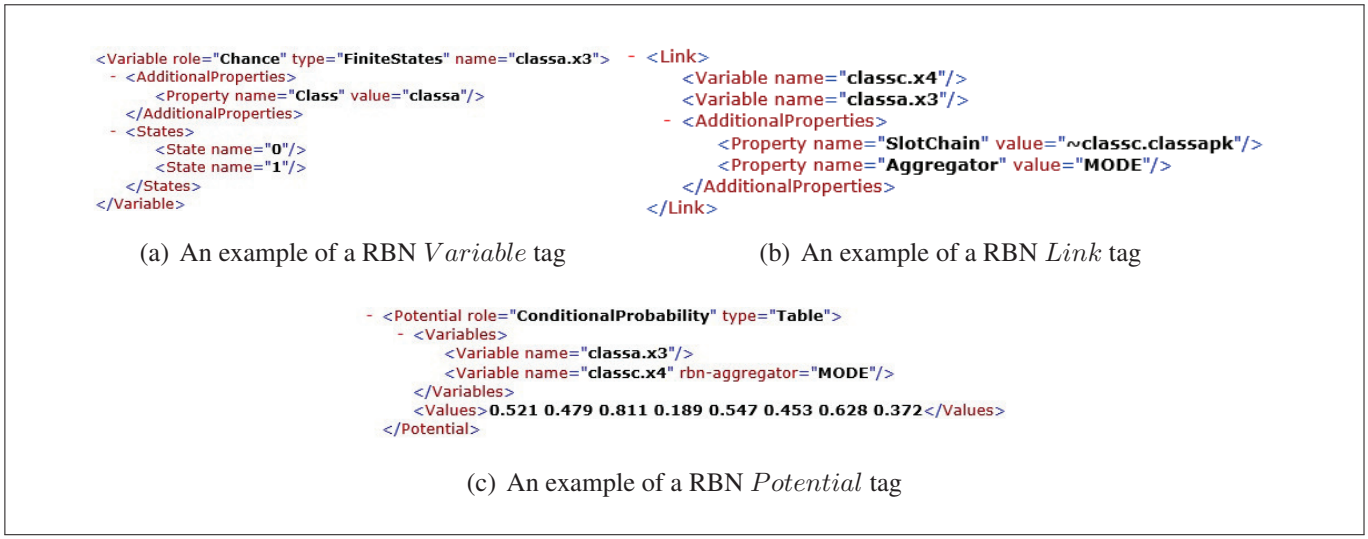
7. http://www.openmarkov.org/



(a) An example of a rbn-Classes tag

(b) An example of a rbn-ClassesLinks tag

Figure 6.5: Additional RBN network properties: The encoded RBN contains 3 classes (Figure 6.5(a)) and 2 reference slots (Figure 6.5(b))



(a) An example of a RBN $Variable$ tag

(b) An example of a RBN $Link$ tag

(c) An example of a RBN $Potential$ tag

Figure 6.6: Main modifications made on already existing properties: In Figure 6.6(a), the variable $classa.x3$ is associated to $classa$. Figure 6.6(b) illustrates an aggregated probabilistic dependency between $classa.x3$ and $classc.x4$. Figure 6.6(c) presents the CPD of $classa.x3$.

known as the maximum likelihood estimation. The Bayesian approach consists on finding the most likely parameters knowing that the data were observed using some prior. As a Bayesian estimator, we can use either the maximum a posteriori (MAP) or the expectation a posteriori (EAP) (Heckerman, 1998). For both approaches, the likelihood function (i.e., the probability of the data given the model) is the main ingredient. For a RBN, the likelihood of a parameter set $\theta_{\mathcal{S}}$ is (Getoor et al., 2007):

$$L(\theta_{\mathcal{S}}|\mathcal{I},\sigma,\mathcal{S}) = P(\mathcal{I}|\sigma,\mathcal{S},\theta_{\mathcal{S}}) \tag{6.1}$$

As usual, we use the log of the likelihood function (Getoor et al., 2007):

$$l(\theta_{\mathcal{S}}|\mathcal{I},\sigma,\mathcal{S}) = logP(\mathcal{I}|\sigma,\mathcal{S},\theta_{\mathcal{S}}) = \sum_{X_i}\sum_{A\in\mathcal{A}(X_i)}\left[\sum_{x\in\sigma(X_i)} log(P(\mathcal{I}_{x.A}|I_{pa_{x.A}}))\right] \tag{6.2}$$

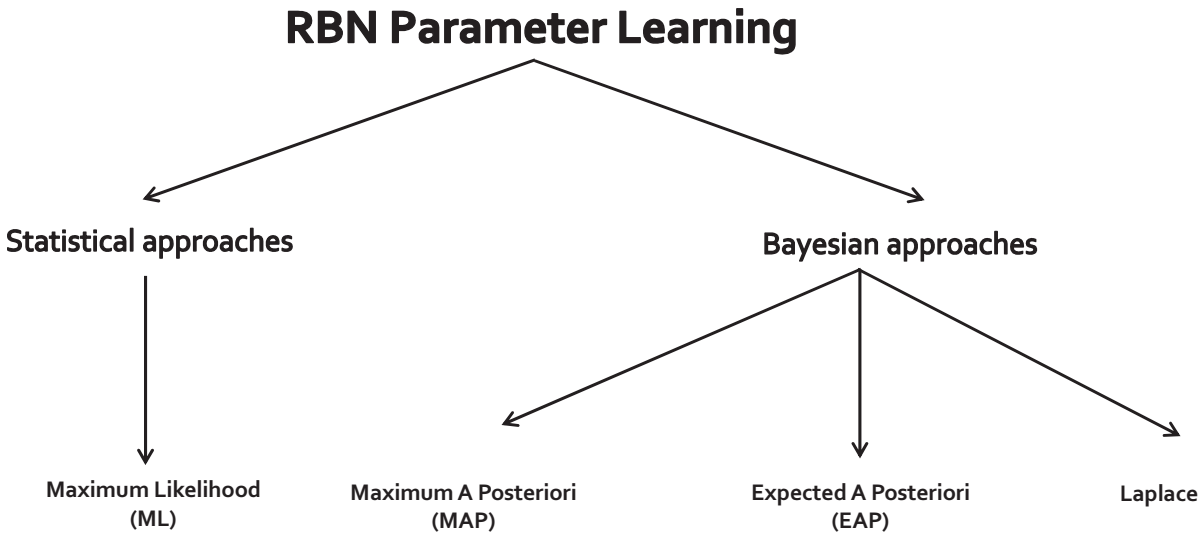


Figure 6.7: Implemented algorithms for parameters learning

PILGRIM Relational provides four possible ways to parameter learning illustrated by Figure 6.7:

– **Maximum Likelihood.** To perform a maximum likelihood estimation of parameters (Getoor et al., 2007):

$$l(\theta_{\mathcal{S}}|\mathcal{I},\sigma,\mathcal{S}) = \sum_{X_i}\sum_{A\in\mathcal{A}(X_i)}\left[\sum_{x\in\sigma(X_i)} log(P(\mathcal{I}_{x.A}|I_{pa_{x.A}}))\right] = \sum_{X_i}\sum_{A\in\mathcal{A}(X_i)}\sum_{v\in V(X_i.A)}\sum_{u\in V(Pa(X_i.A))} C_{X_i.A}[v,u]log\theta_{v|u} \quad (6.3)$$

Where, $C_{X_i.A}[v,u]$, is the number of times we observe each of the different values v, u that the attribute $X_i.A$ and its parents can jointly take.

– **Bayesian approaches.** The Bayesian approach uses a prior distribution over the parameters to deal with the training data irregularities. It is considered significantly more robust than the statistical approach as it reduces the overfiting problem of the likelihood function. Same as the Bayesian approaches for BNs parameter learning (Heckerman, 1998), the prior for a multinomial distribution of a variable $X_i.A$ is a Dirichlet distribution specified by a set of hyper parameters $\alpha[v] : v \in V(X_i.A)$ (Getoor et al., 2007).

  – *Maximum A Posteriori.* Implementing the relational version of the MAP Bayesian approach of parameters learning.

$$MAP_\theta\left[P((X_i.A = v|Pa(X_i.A) = u)|\mathcal{I}\right] = \frac{C_{X_i.A}[v,u] + \alpha_{X_i.A}[v,u] - 1}{\sum_{i=1}^{k}(C_{X_i.A}[v_i,u] + \alpha_{X_i.A}[v_i,u] - 1)} \quad (6.4)$$

  – *Expected A Posteriori.* Implementing the relational version of the EAP Bayesian approach of parameters learning.

$$EAP_\theta\left[P((X_i.A = v|Pa(X_i.A) = u)|\mathcal{I}\right] = \frac{C_{X_i.A}[v,u] + \alpha_{X_i.A}[v,u]}{\sum_{i=1}^{k} C_{X_i.A}[v_i,u] + \alpha_{X_i.A}[v_i,u]} \quad (6.5)$$

  – *Laplace.* Implementing a special case of the Bayesian approach with a uniform prior equal to 1.

6.3.3 Structure learning

This package includes algorithms for RBN structure learning as well as needed score functions and statistical tests illustrated by Figure 6.8.

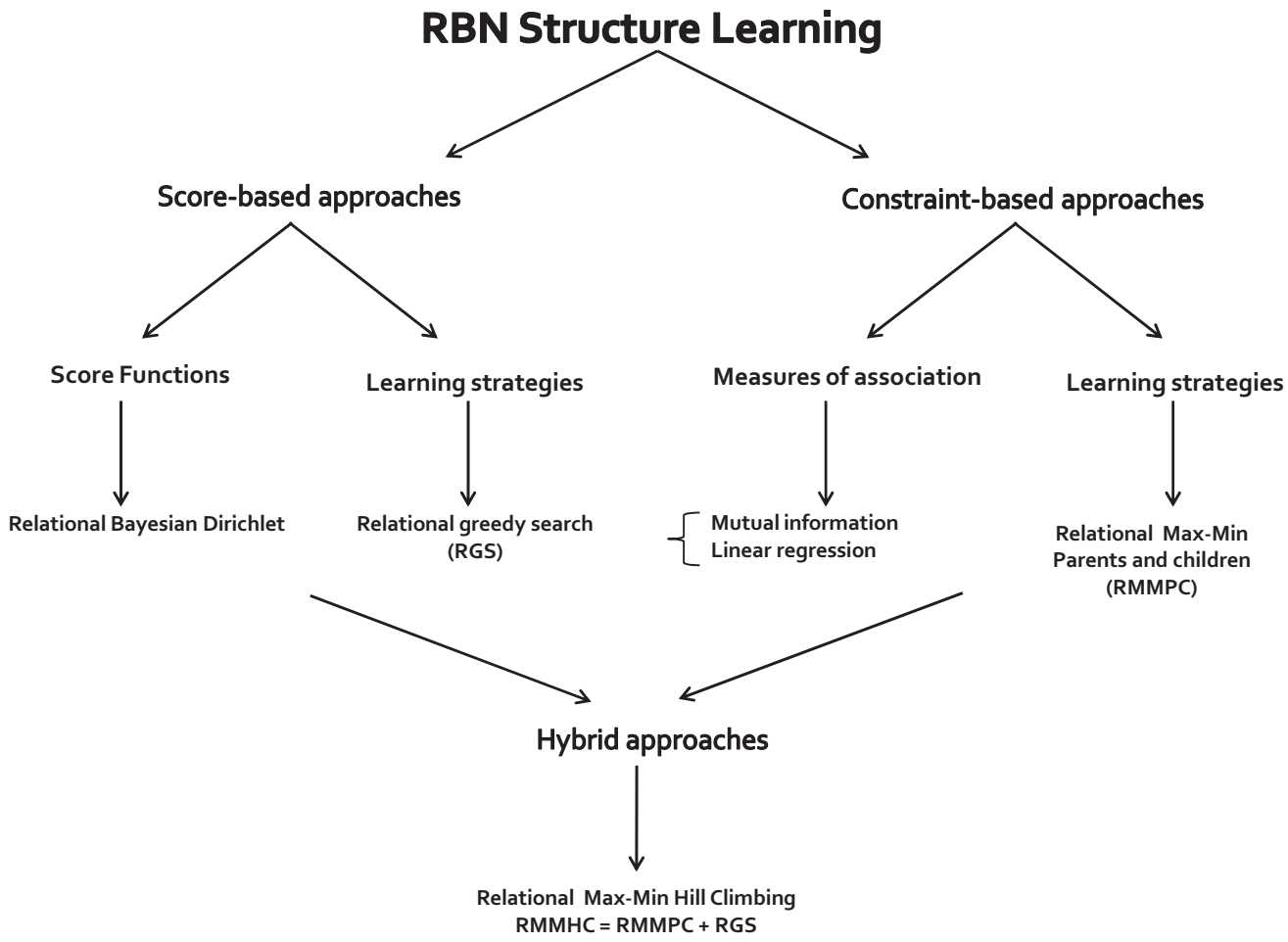


Figure 6.8: Implemented algorithms for structure learning

We have implemented a score-based approach, namely the relational greedy search algorithm with the relational extension of the Bayesian Dirichlet as score function (cf. Section 3.3). We have implemented two variants of the RGS algorithm. The first constructs the neighborhood from all nodes of the RBN. The second constructs the neighborhood from a subset of nodes given as input.

We have also implemented our new hybrid approach of RBN structure learning, the RMMHC algorithm detailed in Chapter 5 with its two variants (cf. Algorithms 27 and 28). During the local search step, the RMMPC algorithm (cf. Algorithm 25) tests statistical independence between variables. To this end, we have implemented two statistical measures of association:

- **The mutual information** (MI) allows to measure the mutual dependence between a set of variables.
- **The linear regression** is a statistical approach for predicting the relationship between a dependent variable and one (simple linear regression) or more (multiple linear regression) explanatory variables. We use the linear regression in our context to measure the dependence between a variable and a set of variables.

During the global search step, the RMMHC algorithm uses the RGS algorithm (cf. Algorithm 14) with as parameters the set of potential parents and children identified by the local search step.

We note that the classical version of the MMHC (cf. Algorithm 8) and GS (cf. Algorithm 3) algorithms is already implemented in PILGRIM structure learning project for BNs. We have used this implementation and extended it to provide the relational version of the algorithms.

6.3.4 RBN structure learning evaluation metrics

We have implemented all the evaluation metrics that we have presented in in Chapter 4. We have implemented our new versions of the Precision, Recall and F-score metrics as well as our new distance-based metric, the RSHD measure, which represents a relational extension of the SHD measure (cf. Section 1.5). All these structure learning evaluation metrics will be used during our experimental study detailed in Chapter 7.

6.3.5 RBN benchmark random generation

To validate implemented learning approaches, we are in need of relational database benchmarks that allow to check probabilistic dependencies among the DB attributes. The benchmark random generation process described in Chapter 4 has been implemented under the PILGRIM relational project. During the implementation, we have used the main policies described in Section 4.2.6.

6.4 Conclusion

In this chapter we have presented the overall organization of the PILGRIM project. Then, we have concentrated on the PILGRIM relational and we have listed our contributions in the development of this software.

The next chapter is dedicated to the experimental protocol for the learning approach. It will serve to prove the effectiveness of the proposed learning approach. In addition, it will present the usefulness of the generation process.

7

Experimental study

After detailing our theoretical contributions in Chapters 4 and 5 and introducing the PILGRIM relational software tool in Chapter 6, this chapter is dedicated to the experimental study regarding the new relational structure learning algorithm. We follow two different experimental protocols where we compare our contribution to state-of-the-art methods with respect to different comparison criteria.

Contents

7.1 Introduction

This chapter is dedicated to the empirical validation of our theoretical contributions. In Section 7.2, we explain our two experiment protocols, for each we will specify used algorithms for the structure learning experiments, used networks as well as the set of evaluation metrics. In Section 7.3, we report the experimental results. We note that all reported results are drawn from experiments carried out on a dedicated PC with Intel(R) Core(TM)i5-3210M CPU 2.5GHz, 64 bits architecture, 6 Gb RAM memory and under Windows 8.1. Finally, in Section 7.4, we made a discussion about the experimentation results, the encountered problems during experimentations and we present some possible solutions to those problems.

7.2 Experimental protocols

As we said previously, only few works have been proposed to learn RBNs and similar model from data, namely RGS,RPC and RCD algorithms (cf. Sections 3.3, 3.3.2). In term of specific implementations, we have re-implemented the RGS algorithm and used our version in the experimental study. RPC and RCD source codes are available respectively in *https://kdl.cs.umass.edu/display/public /Relational+PC* and *https://kdl.cs.umass.edu/display/public/Relational+Causal+Discovery*. The implementation of the RCD algorithm is operational, while the RPC code was difficult to use as it can be used only with a restricted operating system (Mac OS X 10.5) and could not run on our experimental platform. We did not use this algorithm in our comparative study, because, besides the implementation problems, RCD is supposed to correct the theoretical problems of RPC and an experimental study on these two approaches can be found in (Maier et al., 2013a). Consequently, our study includes the following algorithms:

- Relational Greedy Hill-Climbing Search (RGS, re-implemented in our experimental platform using the RBD score, as a score-based approach (cf. Algorithm 14)).
- Relational Causal Discovery (RCD, the available implementation), as a constraint-based approach (cf. Algorithm 15).
- Relational Max-Min Hill-Climbing (RMMHC, our novel hybrid algorithm with both conservative and non conservative versions of the RMMPC algorithm (cf. Sections 5.2) and both single (cf. Algorithm 27) and multiple (cf. Algorithm 28) RGS calls for global optimization. Consequently, our study will include four different versions of the $RMMHC$ algorithm that we refer to as:
  - $RMMHC\_Single\_GS$: The non conservative version of the $RMMHC$ algorithm with single relational greedy search call.
  - $RMMHC^c\_Single\_GS$: The conservative version of the $RMMHC$ algorithm with single relational greedy search call.
  - $RMMHC\_Multiple\_GS$: The non conservative version of the $RMMHC$ algorithm with multiple relational greedy search call.
  - $RMMHC^c\_Multiple\_GS$: The conservative version of the $RMMHC$ algorithm with multiple relational greedy search call.

To compare these algorithms, we need to use them in the same experimental context. As we use an existing code of RCD algorithm, we have to work with the constraints under which RCD could run (see Section 5.5). So we planned two experimental protocols, the first compares RMMHC, RGS and RCD with respect to the RCD implementation requirements, the second compares RMMHC to RGS only, but gives more flexibility in term of generated models, neighborhood generation and used statistical tests.

Network	# entities	# relationships	# K_{max}	# variables	# edges	#states {Min-Max}	#parents {Min-Max}
$\mathcal{RBN}_1$	1	0	0	8	8	$2-2$	$0-2$
$\mathcal{RBN}_2$	2	1	1	8	8	$2-2$	$0-3$
$\mathcal{RBN}_3$	3	2	1	11	7	$2-4$	$0-2$
$\mathcal{RBN}_4$	4	3	2	11	8	$2-3$	$0-2$

Table 7.1: Relational Bayesian networks used by the experimental protocol N°1

7.2.1 RMMHC, RGS, RCD: experimental protocol N°1

7.2.1.1 Specificities using the algorithms

As the RCD algorithm does not support self-relationships (cf. Section 5.5), we provided restricted versions of RGS and RMMHC where self-relationships are prohibited. In addition, as both RCD and RMMHC use statistical independence tests, we have implemented the linear regression test to fit the the RCD implementation and we have used it to perform statistical tests during the local search phase of RMMHC. For judging conditional independence, we have run both RCD and $RMMHC$ using a threshold $\alpha = 0.05$.

7.2.1.2 Networks and datasets

Networks. Unlike standard Bayesian networks, where a set of famous networks is available to perform experimentations (cf. Section 1.5), there is no such models defined in the context of RBNs.

We were not able to use the MovieLens+ database that has been used in (Maier et al., 2013a). It turns out that the licensing of IMDB prohibits the authors from sharing the data. We should mention that there's no ground truth model there, it's merely to show that the algorithm was implemented and it can run on real data. Consequently, we have used our generating process, already described in chapter 4 to generate gold models and relational database instances in order to lunch experimentations.

As RCD implies some constrains on the used network, we have generated networks following their experimental plan described in (Maier et al., 2013a), where:

- Generated schemas contain from 1 to 4 entity classes.
- Generated schemas contain one less than the number of entities as relationship classes.
- Cardinalities are selected uniformly at random
- Attributes per item drawn from $Poison(\lambda = 1) + 1$
- Relational dependencies from 1 to 15, limited by a hop threshold of 4 and at most 3 parents per variable.

Details about all networks included in the study are given in Table 7.1. The K_{max} value in the table refers to the maximum slot chain length found in the generated dependencies of the generated model.

Datasets. For each of the previously described networks, we have randomly sampled 5 relational observational complete datasets with 500, 1000, 2000 and , 3000 instances as an average number of objects per class for each.

7.2.1.3 Evaluation metrics

We have compared the algorithms in term of execution speed and in term of quality of reconstruction. For the former, we have used the number of statistical calls performed by an algorithm as evaluation metric. The quality of reconstruction is measured using RSHD (cf. Algorithm 21), Precision (cf. Formula 4.3), Recall (cf. Formula 4.4) and F-score (cf. Formula 3.10).

Network	# classes	# K_{max}	# variables	# edges	#states {Min-Max}	#parents {Min-Max}
RBN_1	1	0	8	8	$2-2$	$0-2$
RBN_2	2	1	5	4	$2-2$	$0-1$
RBN_3	3	1	8	6	$2-3$	$0-1$
RBN_4	4	2	8	7	$2-3$	$0-2$
RBN_5	5	2	16	10	$2-3$	$0-3$

Table 7.2: Relational Bayesian networks used by the experimental protocol N°2

7.2.2 RMMHC, RGS: experimental protocol N°2

7.2.2.1 Specificities using the algorithms

We will use the extended versions of RGS and RMMHC which support self-relationships. In addition, opposed to the first experimental protocol where we have used linear regression as statistical independence test, the following experimental protocol will use the mutual information association measure with threshold $\alpha = 0.05$. By this way, we will illustrate the implementation of all statistical tests enumerated in Section 6.3.3.

7.2.2.2 Networks and datasets

Networks. We have varied the number of classes from 1 (i.e., the particular case where the RBN collapse to a standard Bayesian network) to 5 and the maximum slot chain length from $K_{max} = 0$ to $K_{max} = 5$. Details about all networks included in the study are given in Table 7.2. The K_{max} value in the table refers to the maximum slot chain length found in the generated dependencies of the generated model.

Datasets. For each of the described networks, we have randomly sampled 5 relational observational complete datasets with 500, 1000, 2000 and 3000 instances as an average number of objects per class.

7.2.2.3 Evaluation metrics

As for the first experimental protocol, we have compared the algorithms in term of execution speed and in term of quality of reconstruction. For the former, we have used the number of statistical calls performed by an algorithm. The quality of reconstruction is measured using both the hard and soft versions of Precision, Recall and F-score defined by Section 4.3.3. However, the RSHD measure is excluded since it has been implemented for the special case where RBNs collapse to DAPER models.

7.2.3 On the choice of the K_{max} value for the learning algorithms

It is clear that any learning algorithm will be unable to find some dependencies if the K_{max} value is lower than the length of slot chains of some dependencies. However,a good learning algorithm should not be very sensitive if the K_{max} is higher than the longest slot chain encountered in the true model. In (Maier et al., 2013a), the authors have fixed a maximum value of 4 as they have fixed the same value for their generated models (cf. Section 7.2.1). For both experimental protocols we fixed K_{max} equal to the maximum slot chain length that really exists in the true model dependencies.

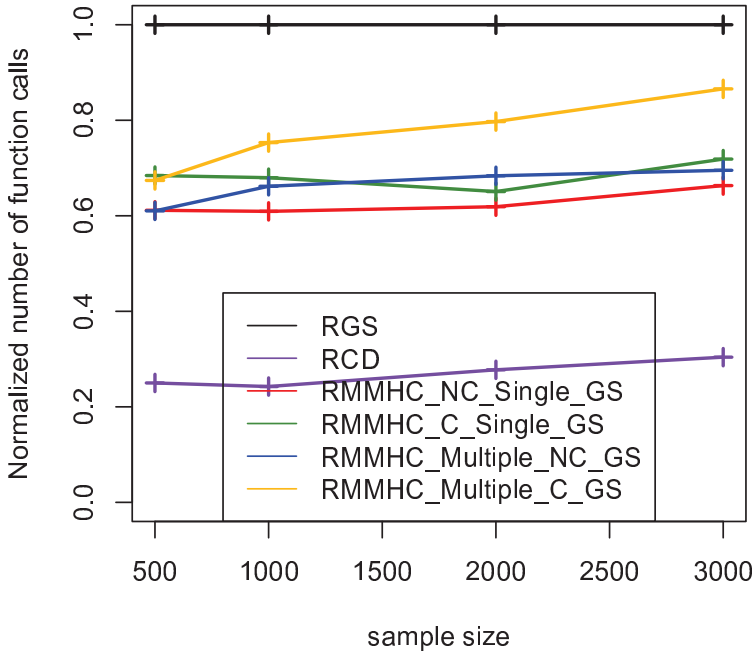


Figure 7.1: Experimental protocol N°1 normalized number of statistical calls with respect to the sample size

7.3 Results and interpretation

7.3.1 Experimental protocol N°1: Results and interpretation

7.3.1.1 Statistical calls results

We are comparing algorithms that are not necessary implemented using the same data structures for representing and performing graph operations. RGS and $RMMHC$ with its different versions are implemented in our local platform while, for the RCD algorithm, we use the available source code.

Table 7.3 contains the normalized number of statistical calls performed by the algorithms we have compared in this experimental protocol. As the used networks are of small size, we have fixed two hours limit to perform computations. Experimentations are canceled when this limit is reached. Also, we note that the RCD algorithm failed to run on some datasets. These latter are also excluded and not reported in Table 7.3 which may explain the low values of RCD. Figure 7.2 illustrates the normalized number of statistical calls with respect to the sample size for all algorithms included in the comparative study.

The RCD algorithm presents better results in term of statistical calls than RGS and RMMHC with its different versions. The $RMMHC\_Single\_GS$ and $RMMHC\_Multiple\_GS$ algorithms performs less statistical calls than $RMMHC^c\_Single\_GS$ and $RMMHC^c\_Multiple\_GS$. This behavior is explained by the fact that the statistical association is computed twice for asymmetric dependencies. In addition, at the symmetrical correction phase, $RMMHC\_Single\_GS$ and $RMMHC\_Multiple\_GS$ algorithms tend to reduce the CPC lists size, whereas, $RMMHC^c\_Single\_GS$ and $RMMHC^c\_Multiple\_GS$ algorithms tend to to expand it which may increase the number of calls of the scoring function during the global optimization phase (cf. Section 5.2.3).

Statistical calls conclusions. The RCD algorithm performs the least number of statistical calls on average, followed by the non conservative versions of the RMMHC algorithm. However, RCD was unscalable for some experimentations while $RMMHC$ scaled with all datasets. In general, the $RMMHC\_Single\_GS$

Model	Algorithm	Data size			
		500	1000	2000	3000
RBN_1	RGS	1.00	1.00	1.00	1.00
	RCD	**0.26**	**0.20**	**0.16**	**0.22**
	$RMMHC\_Single\_GS$	0.66	0.37	0.30	0.48
	$RMMHC^c\_Single\_GS$	0.71	0.42	0.35	0.51
	$RMMHC\_Multiple\_GS$	0.68	0.37	0.30	0.48
	$RMMHC^c\_Multiple\_GS$	0.71	0.42	0.35	0.51
RBN_2	RGS	1.00	1.00	1.00	1.00
	RCD	**0.12**	**0.26**	**0.38**	**0.36**
	$RMMHC\_Single\_GS$	0.15	0.72	0.71	0.74
	$RMMHC^c\_Single\_GS$	0.16	0.76	0.79	0.88
	$RMMHC\_Multiple\_GS$	0.16	0.81	1.02	0.88
	$RMMHC^c\_Multiple\_GS$	0.17	0.94	1.11	1.23
RBN_3	RGS	1.00	1.00	1.00	1.00
	RCD	**0.28**	**0.27**	**0.34**	**0.33**
	$RMMHC\_Single\_GS$	0.90	0.71	0.82	0.78
	$RMMHC^c\_Single\_GS$	0.99	0.78	0.78	0.78
	$RMMHC\_Multiple\_GS$	0.89	0.87	0.88	0.88
	$RMMHC^c\_Multiple\_GS$	0.96	0.95	0.98	0.98
RBN_4	RGS	1.00	1.00	1.00	1.00
	RCD	**0.34**	**0.24**	**0.23**	**0.31**
	$RMMHC\_Single\_GS$	0.75	0.64	0.65	0.66
	$RMMHC^c\_Single\_GS$	0.87	0.77	0.69	0.71
	$RMMHC\_Multiple\_GS$	0.74	0.60	0.54	0.55
	$RMMHC^c\_Multiple\_GS$	0.86	0.70	0.78	0.74
AVG	RGS	1.00	1.00	1.00	1.00
	RCD	**0.25**	**0.24**	**0.28**	**0.30**
	$RMMHC\_Single\_GS$	0.61	0.61	0.62	0.66
	$RMMHC^c\_Single\_GS$	0.68	0.68	0.65	0.72
	$RMMHC\_Multiple\_GS$	0.61	0.66	0.68	0.69
	$RMMHC^c\_Multiple\_GS$	0.67	0.75	0.80	0.87

Table 7.3: Experimental protocol N°1 normalized number of statistical calls (i.e., number of tests of conditional independence and/or number of calls to the local scoring function) performed by each algorithm for a particular sample size and network divided by RGS's calls on the same dataset. Average normalized values lower to one correspond to an algorithm performing less statistical calls than RGS.

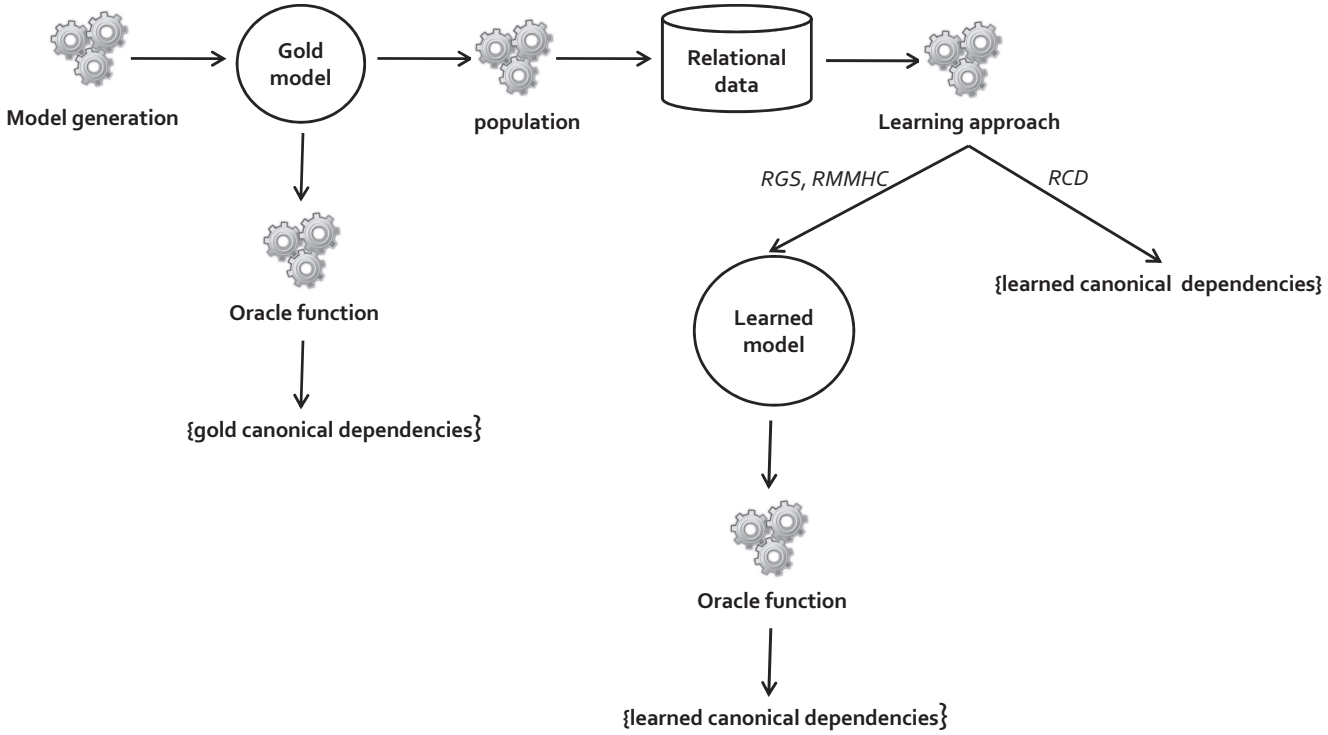


Figure 7.2: Mapping dependency structure into canonical dependencies

performs less statistical calls than $RMMHC\_Multiple\_GS$, except for sample size 500. Also, the $RMMHC^c\_Sin$ performs less statistical calls than $RMMHC^c\_Multiple\_GS$, except for sample size 500.

7.3.1.2 Quality of reconstruction results

In this section, we are comparing algorithms which do not provide the same output. RCD returns a set of canonical dependencies, whereas, RGS and $RMMHC$ return a learned dependency structure $\mathcal{S}$. In order to compare them, we must first standardize their results. To this end, we have used the oracle function available with the RCD source code. This function allows to provide the list of all canonical dependencies of a RBN given as input. Thus, as illustrated by Figure 7.2, we will use the oracle function either to transform the gold model into its set of canonical dependencies or to transform models learned by RGS or $RMMHC$ into their corresponding canonical sets. Evaluation metrics, namely Precision, Recall, F-Measure, and RSHD are applied to canonical dependencies rather than dependency structures.

Tables 7.4, 7.5, 7.6, 7.7 summarize the average $\pm$ standard deviation of RSHD, Precision, Recall and F-Measure evaluation metrics respectively, for all algorithms included in the experimental protocol N°1. Reported values present the average over 5 runs of the algorithms, on 5 generated DB instances, for each RBN from Table 7.1 and each sample size.

Given the model as input, the oracle function aims to find the set of canonical dependencies given a maximum slot chain value. This function fails if the provided value is insufficient to find the result. For $\mathcal{RBN}_4$, the oracle function was unable to find the canonical set of the gold model. We have progressively augmented the k_{max} value, this augmentation resulted in a high running time of this function. It spent more than one day without providing any result. This led us to stop the execution and to exclude the model $\mathcal{RBN}_4$ from the comparative study results. Also, as the remaining models are of $K_{max} \in \{0, 1\}$, hard and soft precision, recall and F-Measure will be of same values. That's why they are presented just once in the summary tables.

Figures 7.3 and 7.4 illustrate the average values of precision, recall, F-score and RSHD measures with respect to the sample size. In general, RGS presents the worst result for all evaluation measures used in the

Model	Algorithm	Data size			
		500	1000	2000	3000
RBN_1	RGS	8.80 ± 1.00	8.40 ± 0.80	8.40 ± 0.80	3.20 ± 0.70
	RCD	**8.00** ± 0.00	**7.75** ± 0.60	3.80 ± 1.38	2.80 ± 1.00
	$RMMHC\_Single\_GS$	**8.00** ± 0.00	**7.75** ± 0.60	**2.00** ± 0.00	**2.00** ± 0.00
	$RMMHC^c\_Single\_GS$	**8.00** ± 0.00	**7.75** ± 0.60	**2.00** ± 0.00	**2.00** ± 0.00
	$RMMHC\_Multiple\_GS$	9.00 ± 0.80	8.60 ± 0.60	**2.00** ± 0.00	**2.00** ± 0.00
	$RMMHC^c\_Multiple\_GS$	9.00 ± 0.80	8.60 ± 0.60	**2.00** ± 0.00	**2.00** ± 0.00
RBN_2	RGS	12.00 ± 0.00	11.80 ± 0.60	12.00 ± 0.00	12.00 ± 0.00
	RCD	8.40 ± 1.00	**7.00** ± 0.60	**4.00** ± 0.00	**4.00** ± 0.00
	$RMMHC\_Single\_GS$	**8.00** ± 1.20	8.00 ± 1.20	5.00 ± 0.60	5.00 ± 0.60
	$RMMHC^c\_Single\_GS$	**8.00** ± 1.20	8.00 ± 1.20	5.00 ± 0.60	5.00 ± 0.60
	$RMMHC\_Multiple\_GS$	**8.00** ± 1.20	8.00 ± 1.20	5.00 ± 0.60	10.00 ± 1.00
	$RMMHC^c\_Multiple\_GS$	**8.00** ± 1.20	8.00 ± 1.20	5.00 ± 0.60	5.00 ± 0.60
RBN_3	RGS	5.33 ± 1.00	7.50 ± 0.80	7.50 ± 0.80	8.00 ± 1.00
	RCD	5.40 ± 1.60	5.50 ± 0.60	**3.00** ± 0.00	4.00 ± 0.60
	$RMMHC\_Single\_GS$	4.00 ± 1.00	**4.00** ± 0.60	4.00 ± 1.12	**3.00** ± 0.86
	$RMMHC^c\_Single\_GS$	**3.00** ± 0.56	**4.00** ± 0.60	4.00 ± 1.12	**3.00** ± 0.86
	$RMMHC\_Multiple\_GS$	4.00 ± 1.00	**4.00** ± 0.60	3.50 ± 0.80	4.00 ± 1.00
	$RMMHC^c\_Multiple\_GS$	4.00 ± 1.00	**4.00** ± 0.60	**3.00** ± 0.56	4.00 ± 1.00
AVG	RGS	8.71	9.23	9.30	7.73
	RCD	7.26	6.75	3.60	3.60
	$RMMHC\_Single\_GS$	6.67	**6.58**	3.67	**3.33**
	$RMMHC^c\_Single\_GS$	**6.33**	**6.58**	3.66	**3.33**
	$RMMHC\_Multiple\_GS$	7.00	6.86	3.50	5.33
	$RMMHC^c\_Multiple\_GS$	7.00	6.87	**3.33**	3.67

Table 7.4: Experimental protocol N°1 Average $\pm$ standard deviation of RSHD for each algorithm for a particular sample size and network. Bold values present the best values for a given model and a given sample size. The AVG values present the average RSHD values for all models for a given sample size.

Model	Algorithm	Data size			
		500	1000	2000	3000
RBN_1	RGS	**0.40** $\pm$ 0.09	0.46 $\pm$ 0.10	0.49 $\pm$ 0.10	0.88 $\pm$ 0.05
	RCD	0.13 $\pm$ 0.10	**0.60** $\pm$ 0.06	0.83 $\pm$ 0.08	0.93 $\pm$ 0.06
	$RMMHC\_Single\_GS$	0.13 $\pm$ 0.10	**0.60** $\pm$ 0.06	**1.00** $\pm$ 0.00	**1.00** $\pm$ 0.00
	$RMMHC^c\_Single\_GS$	0.13 $\pm$ 0.10	**0.60** $\pm$ 0.06	**1.00** $\pm$ 0.00	**1.00** $\pm$ 0.00
	$RMMHC\_Multiple\_GS$	0.37 $\pm$ 0.13	0.44 $\pm$ 0.11	**1.00** $\pm$ 0.00	**1.00** $\pm$ 0.00
	$RMMHC^c\_Multiple\_GS$	0.39 $\pm$ 0.16	0.44 $\pm$ 0.11	**1.00** $\pm$ 0.00	**1.00** $\pm$ 0.00
RBN_2	RGS	0.14 $\pm$ 0.11	0.22 $\pm$ 0.08	0.31 $\pm$ 0.06	0.31 $\pm$ 0.06
	RCD	0.15 $\pm$ 0.14	**0.33** $\pm$ 0.10	**0.69** $\pm$ 0.08	**0.71** $\pm$ 0.06
	$RMMHC\_Single\_GS$	**0.50** $\pm$ 0.10	**0.33** $\pm$ 0.10	0.67 $\pm$ 0.08	0.67 $\pm$ 0.08
	$RMMHC^c\_Single\_GS$	0.33 $\pm$ 0.15	0.17 $\pm$ 0.10	0.67 $\pm$ 0.08	0.67 $\pm$ 0.08
	$RMMHC\_Multiple\_GS$	0.33 $\pm$ 0.15	0.17 $\pm$ 0.10	0.67 $\pm$ 0.10	0.30 $\pm$ 0.16
	$RMMHC^c\_Multiple\_GS$	**0.50** $\pm$ 0.10	**0.33** $\pm$ 0.10	0.67 $\pm$ 0.08	0.67 $\pm$ 0.08
RBN_3	RGS	0.66 $\pm$ 0.08	0.57 $\pm$ 0.11	0.59 $\pm$ 0.06	0.56 $\pm$ 0.06
	RCD	0.69 $\pm$ 0.10	**0.73** $\pm$ 0.09	0.63 $\pm$ 0.11	**0.75** $\pm$ 0.00
	$RMMHC\_Single\_GS$	0.73 $\pm$ 0.09	**0.73** $\pm$ 0.09	0.73 $\pm$ 0.12	0.72 $\pm$ 0.11
	$RMMHC^c\_Single\_GS$	**0.80** $\pm$ 0.05	**0.73** $\pm$ 0.09	0.73 $\pm$ 0.09	0.72 $\pm$ 0.11
	$RMMHC\_Multiple\_GS$	0.73 $\pm$ 0.11	**0.73** $\pm$ 0.09	**0.80** $\pm$ 0.05	0.72 $\pm$ 0.10
	$RMMHC^c\_Multiple\_GS$	0.73 $\pm$ 0.11	**0.73** $\pm$ 0.09	**0.80** $\pm$ 0.05	0.73 $\pm$ 0.10
AVG	RGS	0.40	0.42	0.46	0.58
	RCD	0.32	**0.55**	0.77	**0.80**
	$RMMHC\_Single\_GS$	0.45	**0.55**	0.80	**0.80**
	$RMMHC^c\_Single\_GS$	0.42	0.50	0.80	**0.80**
	$RMMHC\_Multiple\_GS$	0.48	0.46	**0.82**	0.67
	$RMMHC^c\_Multiple\_GS$	**0.53**	0.50	**0.82**	**0.80**

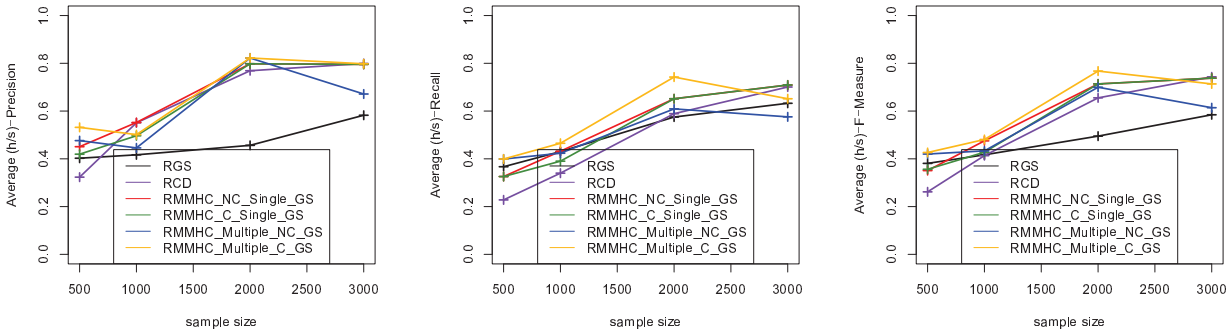
Table 7.5: Experimental protocol N°1 Average $\pm$ standard deviation of Precision for each algorithm for a particular sample size and network. Bold values present the best values for a given model and a given sample size. The AVG values present the average Precision values for all models for a given sample size.

Model	Algorithm	Data size			
		500	1000	2000	3000
RBN_1	RGS	0.31 ± 0.10	0.40 ± 0.00	0.51 ± 0.06	0.67 ± 0.02
	RCD	0.13 ± 0.08	0.32 ± 0.10	0.45 ± 0.05	**0.73** ± 0.06
	$RMMHC\_Single\_GS$	0.13 ± 0.08	0.32 ± 0.10	**0.73** ± 0.06	**0.73** ± 0.06
	$RMMHC^c\_Single\_GS$	0.13 ± 0.08	0.32 ± 0.10	**0.73** ± 0.06	**0.73** ± 0.06
	$RMMHC\_Multiple\_GS$	**0.35** ± 0.10	**0.42** ± 0.08	**0.73** ± 0.06	**0.73** ± 0.06
	$RMMHC^c\_Multiple\_GS$	**0.35** ± 0.10	**0.42** ± 0.08	**0.73** ± 0.06	**0.73** ± 0.06
RBN_2	RGS	**0.13** ± 0.11	**0.30** ± 0.13	0.63 ± 0.11	**0.63** ± 0.06
	RCD	0.05 ± 0.00	0.17 ± 0.09	**0.69** ± 0.10	**0.63** ± 0.06
	$RMMHC\_Single\_GS$	**0.13** ± 0.10	0.25 ± 0.12	0.50 ± 0.00	0.50 ± 0.00
	$RMMHC^c\_Single\_GS$	**0.13** ± 0.10	0.13 ± 0.07	0.50 ± 0.00	0.50 ± 0.00
	$RMMHC\_Multiple\_GS$	**0.13** ± 0.10	0.13 ± 0.10	0.50 ± 0.00	0.38 ± 0.09
	$RMMHC^c\_Multiple\_GS$	**0.13** ± 0.10	0.25 ± 0.11	0.50 ± 0.00	0.50 ± 0.00
RBN_3	RGS	0.67 ± 0.06	0.59 ± 0.12	0.59 ± 0.10	0.60 ± 0.10
	RCD	0.51 ± 0.07	0.55 ± 0.10	0.70 ± 0.05	0.75 ± 0.00
	$RMMHC\_Single\_GS$	**0.73** ± 0.08	**0.73** ± 0.10	0.73 ± 0.10	**0.90** ± 0.05
	$RMMHC^c\_Single\_GS$	**0.73** ± 0.08	**0.73** ± 0.10	0.73 ± 0.10	**0.90** ± 0.05
	$RMMHC\_Multiple\_GS$	**0.73** ± 0.08	**0.73** ± 0.10	0.6 ± 0.15	0.63 ± 0.10
	$RMMHC^c\_Multiple\_GS$	**0.73** ± 0.08	**0.73** ± 0.08	**1.00** ± 0.00	0.73 ± 0.06
AVG	RGS	0.37	0.43	0.58	0.63
	RCD	0.23	0.34	0.59	0.70
	$RMMHC\_Single\_GS$	0.33	0.43	0.65	**0.71**
	$RMMHC^c\_Single\_GS$	0.33	0.39	0.65	**0.71**
	$RMMHC\_Multiple\_GS$	**0.40**	0.42	0.61	0.56
	$RMMHC^c\_Multiple\_GS$	**0.40**	**0.47**	**0.74**	0.65

Table 7.6: Experimental protocol N°1 Average $\pm$ standard deviation of Recall for each algorithm for a particular sample size and network. Bold values present the best values for a given model and a given sample size. The AVG values present the average Recall values for all models for a given sample size.

Model	Algorithm	Data size			
		500	1000	2000	3000
RBN_1	RGS	0.34 ± 0.09	0.42 ± 0.04	0.49 ± 0.07	0.76 ± 0.03
	RCD	0.13 ± 0.10	0.41 ± 0.07	0.58 ± 0.06	0.81 ± 0.06
	$RMMHC\_Single\_GS$	0.13 ± 0.10	0.41 ± 0.07	**0.84** ± 0.04	**0.84** ± 0.04
	$RMMHC^c\_Single\_GS$	0.13 ± 0.10	0.41 ± 0.07	**0.84** ± 0.04	**0.84** ± 0.04
	$RMMHC\_Multiple\_GS$	**0.35** ± 0.11	**0.43** ± 0.10	**0.84** ± 0.04	**0.84** ± 0.04
	$RMMHC^c\_Multiple\_GS$	**0.35** ± 0.14	**0.43** ± 0.10	**0.84** ± 0.04	**0.84** ± 0.04
RBN_2	RGS	0.13 ± 0.11	0.25 ± 0.11	0.42 ± 0.08	0.42 ± 0.06
	RCD	0.07 ± 0.13	0.20 ± 0.09	**0.69** ± 0.09	**0.67** ± 0.06
	$RMMHC\_Single\_GS$	**0.20** ± 0.10	**0.29** ± 0.11	0.57 ± 0.07	0.57 ± 0.07
	$RMMHC^c\_Single\_GS$	0.18 ± 0.12	0.14 ± 0.08	0.57 ± 0.07	0.57 ± 0.07
	$RMMHC\_Multiple\_GS$	0.18 ± 0.13	0.14 ± 0.10	0.57 ± 0.08	0.33 ± 0.12
	$RMMHC^c\_Multiple\_GS$	**0.20** ± 0.10	**0.29** ± 0.10	0.57 ± 0.07	0.57 ± 0.07
RBN_3	RGS	0.66 ± 0.07	0.58 ± 0.11	0.58 ± 0.07	0.58 ± 0.07
	RCD	0.59 ± 0.08	0.62 ± 0.09	0.70 ± 0.07	0.75 ± 0.00
	$RMMHC\_Single\_GS$	0.73 ± 0.08	**0.73** ± 0.09	0.73 ± 0.11	**0.80** ± 0.08
	$RMMHC^c\_Single\_GS$	**0.76** ± 0.06	**0.73** ± 0.09	0.73 ± 0.09	**0.80** ± 0.08
	$RMMHC\_Multiple\_GS$	0.73 ± 0.09	**0.73** ± 0.09	0.69 ± 0.08	0.67 ± 0.10
	$RMMHC^c\_Multiple\_GS$	0.73 ± 0.09	**0.73** ± 0.08	**0.89** ± 0.04	0.73 ± 0.07
AVG	RGS	0.38	0.42	0.50	0.58
	RCD	0.26	0.41	0.66	**0.74**
	$RMMHC\_Single\_GS$	0.35	**0.48**	0.71	**0.74**
	$RMMHC^c\_Single\_GS$	0.36	0.43	0.71	**0.74**
	$RMMHC\_Multiple\_GS$	0.42	0.43	0.70	0.61
	$RMMHC^c\_Multiple\_GS$	**0.43**	**0.48**	**0.77**	0.71

Table 7.7: Experimental protocol N°1 Average $\pm$ standard deviation of F-Measure for each algorithm for a particular sample size and network. Bold values present the best values for a given model and a given sample size. The AVG values present the average F-Measure values for all models for a given sample size.



(a) Average precision (b) Average recall (c) Average F-measure

Figure 7.3: The average values of Precision, Recall and F-Measure with respect to the sample size

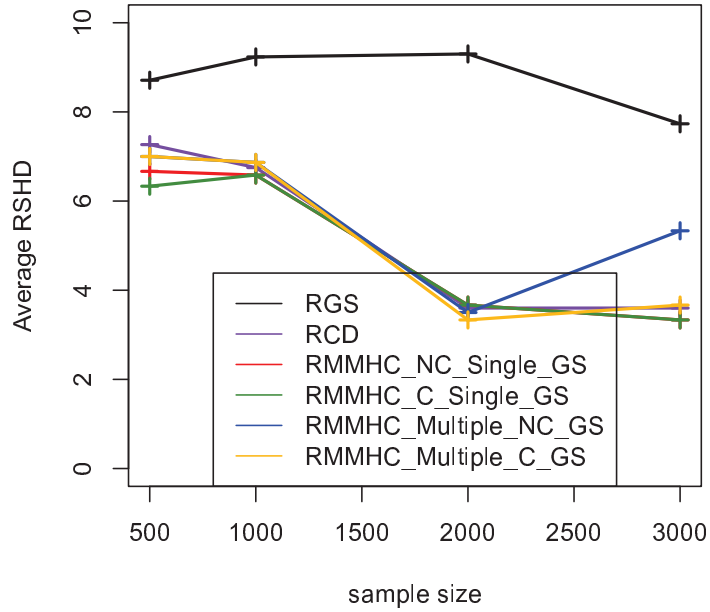


Figure 7.4: The average value of RSHD measure with respect to the sample size

study. RCD and $RMMHC$ with its different versions have very similar results. Precision, recall and F-score metrics are not well suitable to judge the quality of reconstruction of a structure learning algorithms. They may provide some useful interpretation yet, it is preferable to use them with other measurements. In Figure 7.3 RGS outperforms RCD in term of precision, recall and F-score for sample size 500. However, Figure 7.4 shows that for this same sample size, RCD provides better learned structure than RGS as it has a lower $RSHD$ value. For some datasets, the conservative versions of the $RMMHC$ algorithm provide slightly better results than the non conservative ones.

Quality of reconstruction conclusions. The RSHD measure reflects the quality of reconstruction of a learning algorithm better than precision, recall and F-score. In general, RGS presents the worst result in term of RSHD. Even if RCD and $RMMHC$, with its different versions, provide similar results in term of quality of reconstruction, RCD is more efficient as it was able to achieve these quality results with a lowest number of statistical calls. Nevertheless, it was unscalable for some experimentations while $RMMHC$

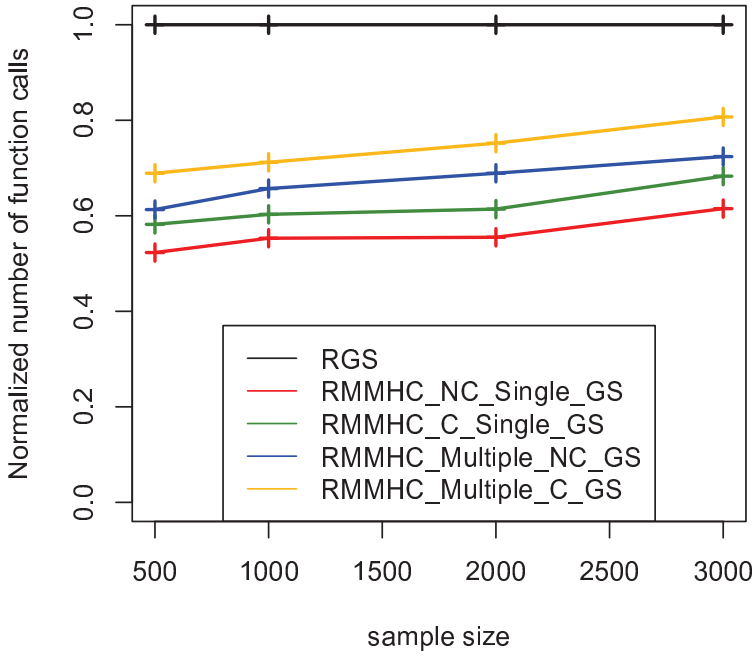


Figure 7.5: Experimental protocol N°2 normalized number of statistical calls with respect to the sample size

scaled with all datasets. In addition, we have used a limited range of models (cf. Section 7.2.1) in order to include RCD in the comparative study.

7.3.2 Experimental protocol N°2: Results and interpretation

7.3.2.1 Statistical calls results

Table 7.8 contains the normalized number of statistical calls performed for the algorithms we have compared in this experimental protocol. As for the first protocol, we have fixed two hours limit to perform computations. Experimentations are canceled when this limit is reached. Figure 7.5 illustrates the normalized number of statistical calls with respect to the sample size for all algorithms included in the comparative study. All $RMMHC$ versions perform less statistical test calls than the RGS algorithm. As for the first experimental protocol, $RMMHC\_Single\_GS$ and $RMMHCc\_Multiple\_GS$ perform less statistical test calls than $RMMHC^c\_Single\_GS$ and $RMMHC^c\_Multiple\_GS$ respectively.

Statistical calls conclusions. All $RMMHC$ versions apply less statistical test calls than RGS algorithm. the $RMMHC\_Single\_GS$ performed the smallest number of tests followed by its non conservative version. The $RMMHC^c\_Multiple\_GS$ performed the highest number of statistical calls among all $RMMHC$ versions.

7.3.2.2 Quality of reconstruction results

Tables 7.9, 7.10, 7.11, 7.12, 7.13 and 7.14 summarize the average $\pm$ standard deviation of hard precision (h-Precision), hard recall (h-Recall), hard F-score (h-F-Measure), soft precision (s-Precision), soft recall (s-Recall) and soft F-score (s-F-Measure) respectively for all the algorithms of the experimental protocol N°2. Reported values present the average over 5 runs of the algorithms, on 5 generated DB instances, for each RBN from Table 7.1 and each sample size. The $RMMHC^c\_Multiple\_GS$ provides better results than

Model	Algorithm	Data size			
		500	1000	2000	3000
RBN_1	RGS	1.00	1.00	1.00	1.00
	$RMMHC\_Single\_GS$	**0.38**	**0.40**	**0.35**	**0.49**
	$RMMHC^c\_Single\_GS$	0.40	0.44	0.37	0.52
	$RMMHC\_Multiple\_GS$	**0.38**	**0.40**	**0.35**	**0.49**
	$RMMHC^c\_Multiple\_GS$	0.40	0.44	0.37	0.52
RBN_2	RGS	1.00	1.00	1.00	1.00
	$RMMHC\_Single\_GS$	**0.56**	**0.59**	**0.63**	**0.62**
	$RMMHC^c\_Single\_GS$	0.58	0.61	0.65	0.64
	$RMMHC\_Multiple\_GS$	0.72	0.82	0.87	0.86
	$RMMHC^c\_Multiple\_GS$	0.73	0.83	0.87	0.86
RBN_3	RGS	1.00	1.00	1.00	1.00
	$RMMHC\_Single\_GS$	**0.61**	**0.55**	**0.61**	**0.67**
	$RMMHC^c\_Single\_GS$	0.66	0.59	0.68	0.77
	$RMMHC\_Multiple\_GS$	0.77	0.72	0.81	0.81
	$RMMHC^c\_Multiple\_GS$	0.81	0.76	0.86	0.96
RBN_4	RGS	1.00	1.00	1.00	1.00
	$RMMHC\_Single\_GS$	**0.82**	**0.81**	**0.71**	**0.80**
	$RMMHC^c\_Single\_GS$	0.89	0.90	0.80	0.91
	$RMMHC\_Multiple\_GS$	0.87	0.90	0.85	0.81
	$RMMHC^c\_Multiple\_GS$	0.94	1.00	0.96	0.94
RBN_5	RGS	1.00	1.00	1.00	1.00
	$RMMHC\_Single\_GS$	**0.31**	**0.41**	**0.47**	**0.50**
	$RMMHC^c\_Single\_GS$	0.38	0.49	0.59	0.58
	$RMMHC\_Multiple\_GS$	0.33	0.45	0.57	0.65
	$RMMHC^c\_Multiple\_GS$	0.42	0.54	0.69	0.75
AVG	RGS	1.00	1.00	1.00	1.00
	$RMMHC\_Single\_GS$	**0.52**	**0.55**	**0.56**	**0.62**
	$RMMHC^c\_Single\_GS$	0.58	0.60	0.61	0.68
	$RMMHC\_Multiple\_GS$	0.61	0.66	0.69	0.72
	$RMMHC^c\_Multiple\_GS$	0.69	0.71	0.75	0.81

Table 7.8: Experimental protocol N°2 normalized number of statistical calls (i.e., number of tests of conditional independence and/or number of calls to the local scoring function) performed by each algorithm for a particular sample size and network divided by RGS's calls on the same dataset. Average normalized values lower to one correspond to an algorithm performing less statistical calls than RGS.

Model	Algorithm	Data size			
		500	1000	2000	3000
RBN_1	RGS	0.25 ± 0.14	**0.22** ± 0.11	0.28 ± 0.15	0.79 ± 0.07
	$RMMHC\_Single\_GS$	**0.26** ± 0.15	0.21 ± 0.14	0.32 ± 0.14	0.79 ± 0.05
	$RMMHC^c\_Single\_GS$	**0.26** ± 0.15	**0.22** ± 0.14	**0.33** ± 0.14	**0.81** ± 0.05
	$RMMHC\_Multiple\_GS$	**0.26** ± 0.15	0.21 ± 0.14	0.32 ± 0.14	0.79 ± 0.05
	$RMMHC^c\_Multiple\_GS$	**0.26** ± 0.15	**0.22** ± 0.14	**0.33** ± 0.14	**0.81** ± 0.05
RBN_2	RGS	0.56 ± 0.10	0.65 ± 0.04	0.75 ± 0.03	0.75 ± 0.05
	$RMMHC\_Single\_GS$	0.55 ± 0.11	**0.72** ± 0.09	**0.80** ± 0.04	**0.80** ± 0.05
	$RMMHC^c\_Single\_GS$	0.55 ± 0.11	**0.72** ± 0.09	**0.80** ± 0.04	**0.80** ± 0.05
	$RMMHC\_Multiple\_GS$	**0.57** ± 0.11	**0.72** ± 0.09	**0.80** ± 0.04	**0.80** ± 0.05
	$RMMHC^c\_Multiple\_GS$	**0.57** ± 0.11	**0.72** ± 0.09	**0.80** ± 0.04	**0.80** ± 0.05
RBN_3	RGS	0.32 ± 0.17	0.37 ± 0.13	0.39 ± 0.07	0.49 ± 0.15
	$RMMHC\_Single\_GS$	**0.40** ± 0.10	**0.40** ± 0.12	**0.46** ± 0.09	**0.50** ± 0.12
	$RMMHC^c\_Single\_GS$	**0.40** ± 0.10	**0.40** ± 0.12	**0.46** ± 0.09	**0.50** ± 0.12
	$RMMHC\_Multiple\_GS$	0.37 ± 0.11	**0.40** ± 0.12	**0.46** ± 0.09	**0.50** ± 0.12
	$RMMHC^c\_Multiple\_GS$	**0.40** ± 0.10	**0.40** ± 0.12	**0.46** ± 0.09	**0.50** ± 0.12
RBN_4	RGS	0.27 ± 0.15	0.27 ± 0.19	0.29 ± 0.17	0.30 ± 0.11
	$RMMHC\_Single\_GS$	**0.30** ± 0.17	**0.43** ± 0.13	**0.31** ± 0.17	**0.32** ± 0.12
	$RMMHC^c\_Single\_GS$	**0.30** ± 0.17	**0.43** ± 0.13	**0.31** ± 0.17	**0.32** ± 0.12
	$RMMHC\_Multiple\_GS$	0.29 ± 0.17	0.40 ± 0.14	**0.31** ± 0.17	**0.32** ± 0.12
	$RMMHC^c\_Multiple\_GS$	0.29 ± 0.17	0.40 ± 0.14	**0.31** ± 0.17	**0.32** ± 0.12
RBN_5	RGS	0.32 ± 0.11	0.31 ± 0.13	0.59 ± 0.09	0.59 ± 0.09
	$RMMHC\_Single\_GS$	0.65 ± 0.05	**0.36** ± 0.10	**0.71** ± 0.02	**0.63** ± 0.04
	$RMMHC^c\_Single\_GS$	**0.69** ± 0.04	**0.36** ± 0.10	**0.71** ± 0.02	**0.63** ± 0.04
	$RMMHC\_Multiple\_GS$	0.59 ± 0.07	**0.36** ± 0.10	**0.71** ± 0.02	**0.63** ± 0.04
	$RMMHC^c\_Multiple\_GS$	0.62 ± 0.06	**0.36** ± 0.10	**0.71** ± 0.02	**0.63** ± 0.04
AVG	RGS	0.35	0.36	0.46	0.58
	$RMMHC\_Single\_GS$	0.40	**0.42**	**0.52**	**0.61**
	$RMMHC^c\_Single\_GS$	**0.44**	**0.42**	**0.52**	**0.61**
	$RMMHC\_Multiple\_GS$	0.42	**0.42**	**0.52**	**0.61**
	$RMMHC^c\_Multiple\_GS$	0.43	**0.42**	**0.52**	**0.61**

Table 7.9: Experimental protocol N°2 Average $\pm$ standard deviation of h-Precision for each algorithm for a particular sample size and network. Bold values present the best values for a given model and a given sample size. The AVG values present the average h-Precision values for all models for a given sample size.

Model	Algorithm	Data size			
		500	1000	2000	3000
RBN_1	RGS	**0.13** $\pm$ 0.03	**0.13** $\pm$ 0.03	**0.23** $\pm$ 0.10	**0.63** $\pm$ 0.01
	$RMMHC\_Single\_GS$	**0.13** $\pm$ 0.03	**0.13** $\pm$ 0.03	**0.23** $\pm$ 0.10	**0.63** $\pm$ 0.01
	$RMMHC^c\_Single\_GS$	**0.13** $\pm$ 0.03	**0.13** $\pm$ 0.03	**0.23** $\pm$ 0.10	**0.63** $\pm$ 0.01
	$RMMHC\_Multiple\_GS$	**0.13** $\pm$ 0.03	**0.13** $\pm$ 0.03	**0.23** $\pm$ 0.10	**0.63** $\pm$ 0.01
	$RMMHC^c\_Multiple\_GS$	**0.13** $\pm$ 0.03	**0.13** $\pm$ 0.03	**0.23** $\pm$ 0.10	**0.63** $\pm$ 0.01
RBN_2	RGS	0.60 $\pm$ 0.07	0.65 $\pm$ 0.02	0.75 $\pm$ 0.00	0.75 $\pm$ 0.00
	$RMMHC\_Single\_GS$	0.55 $\pm$ 0.05	**0.90** $\pm$ 0.00	**1.00** $\pm$ 0.00	**1.00** $\pm$ 0.00
	$RMMHC^c\_Single\_GS$	0.55 $\pm$ 0.05	**0.90** $\pm$ 0.00	**1.00** $\pm$ 0.00	**1.00** $\pm$ 0.00
	$RMMHC\_Multiple\_GS$	**0.65** $\pm$ 0.03	**0.90** $\pm$ 0.00	**1.00** $\pm$ 0.00	**1.00** $\pm$ 0.00
	$RMMHC^c\_Multiple\_GS$	**0.65** $\pm$ 0.03	**0.90** $\pm$ 0.00	**1.00** $\pm$ 0.00	**1.00** $\pm$ 0.00
RBN_3	RGS	0.33 $\pm$ 0.02	**0.40** $\pm$ 0.05	**0.43** $\pm$ 0.05	0.53 $\pm$ 0.03
	$RMMHC\_Single\_GS$	**0.37** $\pm$ 0.04	0.37 $\pm$ 0.04	**0.43** $\pm$ 0.02	**0.63** $\pm$ 0.03
	$RMMHC^c\_Single\_GS$	**0.37** $\pm$ 0.04	0.37 $\pm$ 0.04	**0.43** $\pm$ 0.02	**0.63** $\pm$ 0.03
	$RMMHC\_Multiple\_GS$	0.33 $\pm$ 0.03	0.37 $\pm$ 0.04	**0.43** $\pm$ 0.02	**0.63** $\pm$ 0.03
	$RMMHC^c\_Multiple\_GS$	**0.37** $\pm$ 0.04	0.37 $\pm$ 0.04	**0.43** $\pm$ 0.02	**0.63** $\pm$ 0.03
RBN_4	RGS	**0.23** $\pm$ 0.09	0.26 $\pm$ 0.05	**0.29** $\pm$ 0.07	**0.29** $\pm$ 0.06
	$RMMHC\_Single\_GS$	0.20 $\pm$ 0.10	**0.50** $\pm$ 0.00	**0.29** $\pm$ 0.04	**0.29** $\pm$ 0.02
	$RMMHC^c\_Single\_GS$	0.20 $\pm$ 0.10	**0.50** $\pm$ 0.00	**0.29** $\pm$ 0.04	**0.29** $\pm$ 0.02
	$RMMHC\_Multiple\_GS$	0.20 $\pm$ 0.10	0.40 $\pm$ 0.03	**0.29** $\pm$ 0.04	**0.29** $\pm$ 0.02
	$RMMHC^c\_Multiple\_GS$	0.20 $\pm$ 0.10	0.40 $\pm$ 0.03	**0.29** $\pm$ 0.04	**0.29** $\pm$ 0.02
RBN_5	RGS	**0.50** $\pm$ 0.00	**0.50** $\pm$ 0.00	**1.00** $\pm$ 0.00	**1.00** $\pm$ 0.00
	$RMMHC\_Single\_GS$	**0.50** $\pm$ 0.00	**0.50** $\pm$ 0.00	**1.00** $\pm$ 0.00	**1.00** $\pm$ 0.00
	$RMMHC^c\_Single\_GS$	**0.50** $\pm$ 0.00	**0.50** $\pm$ 0.00	**1.00** $\pm$ 0.00	**1.00** $\pm$ 0.00
	$RMMHC\_Multiple\_GS$	**0.50** $\pm$ 0.00	**0.50** $\pm$ 0.00	**1.00** $\pm$ 0.00	**1.00** $\pm$ 0.00
	$RMMHC^c\_Multiple\_GS$	**0.50** $\pm$ 0.00	**0.50** $\pm$ 0.00	**1.00** $\pm$ 0.00	**1.00** $\pm$ 0.00
AVG	RGS	0.36	0.39	0.54	0.64
	$RMMHC\_Single\_GS$	0.36	**0.48**	**0.59**	**0.71**
	$RMMHC^c\_Single\_GS$	0.35	**0.48**	**0.59**	**0.71**
	$RMMHC\_Multiple\_GS$	0.36	0.46	**0.59**	**0.71**
	$RMMHC^c\_Multiple\_GS$	**0.37**	0.46	**0.59**	**0.71**

Table 7.10: Experimental protocol N°2 Average $\pm$ standard deviation of h-Recall for each algorithm for a particular sample size and network. Bold values present the best values for a given model and a given sample size. The AVG values present the average h-Recall values for all models for a given sample size.

Model	Algorithm	Data size			
		500	1000	2000	3000
RBN_1	RGS	0.16 ± 0.05	**0.16** ± 0.05	0.25 ± 0.12	**0.70** ± 0.02
	$RMMHC\_Single\_GS$	**0.17** ± 0.05	**0.16** ± 0.05	0.26 ± 0.12	**0.70** ± 0.02
	$RMMHC^c\_Single\_GS$	**0.17** ± 0.05	**0.16** ± 0.05	**0.27** ± 0.12	**0.70** ± 0.02
	$RMMHC\_Multiple\_GS$	**0.17** ± 0.05	**0.16** ± 0.05	0.26 ± 0.12	**0.70** ± 0.02
	$RMMHC^c\_Multiple\_GS$	**0.17** ± 0.05	**0.16** ± 0.05	**0.27** ± 0.12	**0.70** ± 0.02
RBN_2	RGS	0.58 ± 0.08	0.65 ± 0.03	0.75 ± 0.00	0.75 ± 0.00
	$RMMHC\_Single\_GS$	0.55 ± 0.07	**0.80** ± 0.07	**0.89** ± 0.04	**0.89** ± 0.04
	$RMMHC^c\_Single\_GS$	0.55 ± 0.07	**0.80** ± 0.07	**0.89** ± 0.04	**0.89** ± 0.04
	$RMMHC\_Multiple\_GS$	**0.61** ± 0.05	**0.80** ± 0.07	**0.89** ± 0.04	**0.89** ± 0.04
	$RMMHC^c\_Multiple\_GS$	**0.61** ± 0.05	**0.80** ± 0.07	**0.89** ± 0.04	**0.89** ± 0.04
RBN_3	RGS	0.33 ± 0.05	**0.38** ± 0.08	0.41 ± 0.06	0.51 ± 0.06
	$RMMHC\_Single\_GS$	**0.38** ± 0.06	**0.38** ± 0.06	**0.45** ± 0.03	**0.56** ± 0.06
	$RMMHC^c\_Single\_GS$	**0.38** ± 0.06	**0.38** ± 0.06	**0.45** ± 0.03	**0.56** ± 0.06
	$RMMHC\_Multiple\_GS$	0.35 ± 0.07	**0.38** ± 0.06	**0.45** ± 0.03	**0.56** ± 0.06
	$RMMHC^c\_Multiple\_GS$	**0.38** ± 0.06	**0.38** ± 0.06	**0.45** ± 0.03	**0.56** ± 0.06
RBN_4	RGS	**0.25** ± 0.12	0.26 ± 0.10	0.29 ± 0.13	0.29 ± 0.07
	$RMMHC\_Single\_GS$	0.24 ± 0.14	**0.46** ± 0.09	**0.30** ± 0.07	**0.30** ± 0.04
	$RMMHC^c\_Single\_GS$	0.24 ± 0.14	**0.46** ± 0.09	**0.30** ± 0.07	**0.30** ± 0.04
	$RMMHC\_Multiple\_GS$	0.23 ± 0.14	0.40 ± 0.06	**0.30** ± 0.07	**0.30** ± 0.04
	$RMMHC^c\_Multiple\_GS$	0.23 ± 0.14	0.40 ± 0.06	**0.30** ± 0.07	**0.30** ± 0.04
RBN_5	RGS	0.40 ± 0.09	0.38 ± 0.10	0.74 ± 0.07	0.74 ± 0.07
	$RMMHC\_Single\_GS$	0.55 ± 0.04	**0.42** ± 0.08	**0.83** ± 0.02	**0.77** ± 0.03
	$RMMHC^c\_Single\_GS$	**0.57** ± 0.03	**0.42** ± 0.08	**0.83** ± 0.02	**0.77** ± 0.03
	$RMMHC\_Multiple\_GS$	0.54 ± 0.05	**0.42** ± 0.08	**0.83** ± 0.02	**0.77** ± 0.03
	$RMMHC^c\_Multiple\_GS$	0.55 ± 0.05	**0.42** ± 0.08	**0.83** ± 0.02	**0.77** ± 0.03
AVG	RGS	0.34	0.37	0.50	0.61
	$RMMHC\_Single\_GS$	0.38	**0.44**	**0.55**	**0.64**
	$RMMHC^c\_Single\_GS$	**0.40**	**0.44**	**0.55**	**0.64**
	$RMMHC\_Multiple\_GS$	0.38	0.43	**0.55**	**0.64**
	$RMMHC^c\_Multiple\_GS$	0.39	0.43	**0.55**	**0.64**

Table 7.11: Experimental protocol N°2 Average $\pm$ standard deviation of h-F-Measure for each algorithm for a particular sample size and network. Bold values present the best values for a given model and a given sample size. The AVG values present the average h-F-Measure values for all models for a given sample size.

Model	Algorithm	Data size			
		500	1000	2000	3000
RBN_1	RGS	0.25 ± 0.14	**0.22** ± 0.11	0.28 ± 0.15	0.79 ± 0.07
	$RMMHC\_Single\_GS$	**0.26** ± 0.15	0.21 ± 0.14	0.32 ± 0.14	0.79 ± 0.05
	$RMMHC^c\_Single\_GS$	**0.26** ± 0.15	**0.22** ± 0.14	**0.33** ± 0.14	**0.81** ± 0.05
	$RMMHC\_Multiple\_GS$	**0.26** ± 0.15	0.21 ± 0.14	0.32 ± 0.14	0.79 ± 0.05
	$RMMHC^c\_Multiple\_GS$	**0.26** ± 0.15	**0.22** ± 0.14	**0.33** ± 0.14	**0.81** ± 0.05
RBN_2	RGS	0.56 ± 0.10	0.65 ± 0.04	0.75 ± 0.03	0.75 ± 0.05
	$RMMHC\_Single\_GS$	0.55 ± 0.11	**0.72** ± 0.09	**0.80** ± 0.04	**0.80** ± 0.05
	$RMMHC^c\_Single\_GS$	0.55 ± 0.11	**0.72** ± 0.09	**0.80** ± 0.04	**0.80** ± 0.05
	$RMMHC\_Multiple\_GS$	**0.57** ± 0.11	**0.72** ± 0.09	**0.80** ± 0.04	**0.80** ± 0.05
	$RMMHC^c\_Multiple\_GS$	**0.57** ± 0.11	**0.72** ± 0.09	**0.80** ± 0.04	**0.80** ± 0.05
RBN_3	RGS	0.32 ± 0.17	0.37 ± 0.13	0.39 ± 0.07	0.49 ± 0.15
	$RMMHC\_Single\_GS$	**0.40** ± 0.10	**0.40** ± 0.12	**0.46** ± 0.09	**0.50** ± 0.12
	$RMMHC^c\_Single\_GS$	**0.40** ± 0.10	**0.40** ± 0.12	**0.46** ± 0.09	**0.50** ± 0.12
	$RMMHC\_Multiple\_GS$	0.37 ± 0.11	**0.40** ± 0.12	**0.46** ± 0.09	**0.50** ± 0.12
	$RMMHC^c\_Multiple\_GS$	**0.40** ± 0.10	**0.40** ± 0.12	**0.46** ± 0.09	**0.50** ± 0.12
RBN_4	RGS	0.27 ± 0.15	0.27 ± 0.19	0.29 ± 0.17	0.30 ± 0.11
	$RMMHC\_Single\_GS$	**0.30** ± 0.17	**0.43** ± 0.13	**0.31** ± 0.17	**0.32** ± 0.12
	$RMMHC^c\_Single\_GS$	**0.30** ± 0.17	**0.43** ± 0.13	**0.31** ± 0.17	**0.32** ± 0.12
	$RMMHC\_Multiple\_GS$	0.29 ± 0.17	0.40 ± 0.14	**0.31** ± 0.17	**0.32** ± 0.12
	$RMMHC^c\_Multiple\_GS$	0.29 ± 0.17	0.40 ± 0.14	**0.31** ± 0.17	**0.32** ± 0.12
RBN_5	RGS	0.32 ± 0.11	0.31 ± 0.13	0.59 ± 0.09	0.59 ± 0.09
	$RMMHC\_Single\_GS$	0.65 ± 0.05	0.36 ± 0.10	**0.71** ± 0.02	**0.63** ± 0.04
	$RMMHC^c\_Single\_GS$	**0.69** ± 0.04	0.36 ± 0.10	**0.71** ± 0.02	**0.63** ± 0.04
	$RMMHC\_Multiple\_GS$	0.59 ± 0.07	**0.55** ± 0.08	**0.71** ± 0.02	**0.63** ± 0.04
	$RMMHC^c\_Multiple\_GS$	0.62 ± 0.06	**0.55** ± 0.08	**0.71** ± 0.02	**0.63** ± 0.04
AVG	RGS	0.35	0.36	0.46	0.58
	$RMMHC\_Single\_GS$	0.40	0.42	**0.52**	**0.61**
	$RMMHC^c\_Single\_GS$	**0.44**	0.42	**0.52**	**0.61**
	$RMMHC\_Multiple\_GS$	0.42	**0.46**	**0.52**	**0.61**
	$RMMHC^c\_Multiple\_GS$	0.43	**0.46**	**0.52**	**0.61**

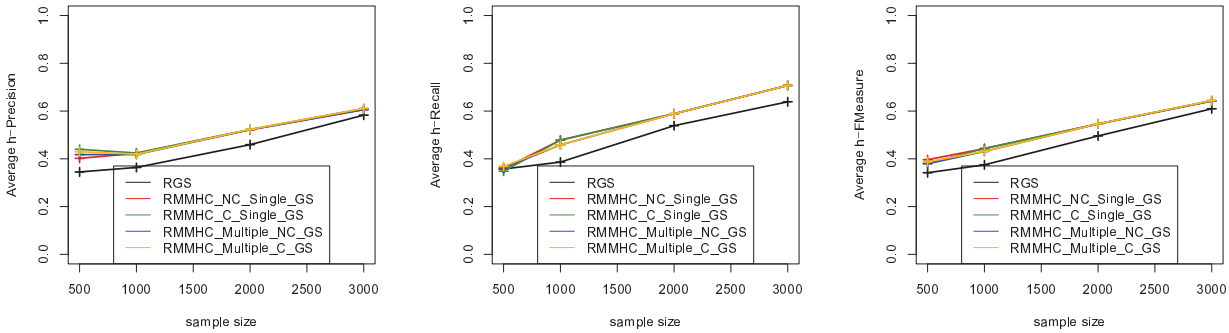
Table 7.12: Experimental protocol N°2 Average $\pm$ standard deviation of s-Precision for each algorithm for a particular sample size and network. Bold values present the best values for a given model and a given sample size. The AVG values present the average s-Precision values for all models for a given sample size.

Model	Algorithm	Data size			
		500	1000	2000	3000
RBN_1	RGS	**0.13** $\pm$ 0.03	**0.13** $\pm$ 0.03	**0.23** $\pm$ 0.10	**0.63** $\pm$ 0.01
	$RMMHC\_Single\_GS$	**0.13** $\pm$ 0.03	**0.13** $\pm$ 0.03	**0.23** $\pm$ 0.10	**0.63** $\pm$ 0.01
	$RMMHC^c\_Single\_GS$	**0.13** $\pm$ 0.03	**0.13** $\pm$ 0.03	**0.23** $\pm$ 0.10	**0.63** $\pm$ 0.01
	$RMMHC\_Multiple\_GS$	**0.13** $\pm$ 0.03	**0.13** $\pm$ 0.03	**0.23** $\pm$ 0.10	**0.63** $\pm$ 0.01
	$RMMHC^c\_Multiple\_GS$	**0.13** $\pm$ 0.03	**0.13** $\pm$ 0.03	**0.23** $\pm$ 0.10	**0.63** $\pm$ 0.01
RBN_2	RGS	0.58 $\pm$ 0.08	0.65 $\pm$ 0.03	0.75 $\pm$ 0.00	0.75 $\pm$ 0.00
	$RMMHC\_Single\_GS$	0.55 $\pm$ 0.07	**0.80** $\pm$ 0.07	**0.89** $\pm$ 0.04	**0.89** $\pm$ 0.04
	$RMMHC^c\_Single\_GS$	0.55 $\pm$ 0.07	**0.80** $\pm$ 0.07	**0.89** $\pm$ 0.04	**0.89** $\pm$ 0.04
	$RMMHC\_Multiple\_GS$	**0.61** $\pm$ 0.05	**0.80** $\pm$ 0.07	**0.89** $\pm$ 0.04	**0.89** $\pm$ 0.04
	$RMMHC^c\_Multiple\_GS$	**0.61** $\pm$ 0.05	**0.80** $\pm$ 0.07	**0.89** $\pm$ 0.04	**0.89** $\pm$ 0.04
RBN_3	RGS	0.33 $\pm$ 0.02	**0.40** $\pm$ 0.05	**0.43** $\pm$ 0.05	0.53 $\pm$ 0.03
	$RMMHC\_Single\_GS$	**0.37** $\pm$ 0.04	0.37 $\pm$ 0.04	**0.43** $\pm$ 0.02	**0.63** $\pm$ 0.03
	$RMMHC^c\_Single\_GS$	**0.37** $\pm$ 0.04	0.37 $\pm$ 0.04	**0.43** $\pm$ 0.02	**0.63** $\pm$ 0.03
	$RMMHC\_Multiple\_GS$	0.33 $\pm$ 0.03	0.37 $\pm$ 0.04	**0.43** $\pm$ 0.02	**0.63** $\pm$ 0.03
	$RMMHC^c\_Multiple\_GS$	**0.37** $\pm$ 0.04	0.37 $\pm$ 0.04	**0.43** $\pm$ 0.02	**0.63** $\pm$ 0.03
RBN_4	RGS	**0.23** $\pm$ 0.09	0.26 $\pm$ 0.05	**0.29** $\pm$ 0.07	**0.29** $\pm$ 0.06
	$RMMHC\_Single\_GS$	0.20 $\pm$ 0.10	**0.50** $\pm$ 0.00	**0.29** $\pm$ 0.04	**0.29** $\pm$ 0.02
	$RMMHC^c\_Single\_GS$	0.20 $\pm$ 0.10	**0.50** $\pm$ 0.00	**0.29** $\pm$ 0.04	**0.29** $\pm$ 0.02
	$RMMHC\_Multiple\_GS$	0.20 $\pm$ 0.10	0.40 $\pm$ 0.03	**0.29** $\pm$ 0.04	**0.29** $\pm$ 0.02
	$RMMHC^c\_Multiple\_GS$	0.20 $\pm$ 0.10	0.40 $\pm$ 0.03	**0.29** $\pm$ 0.04	**0.29** $\pm$ 0.02
RBN_5	RGS	**0.50** $\pm$ 0.00	0.50 $\pm$ 0.00	**1.00** $\pm$ 0.00	**1.00** $\pm$ 0.00
	$RMMHC\_Single\_GS$	**0.50** $\pm$ 0.00	0.50 $\pm$ 0.00	**1.00** $\pm$ 0.00	**1.00** $\pm$ 0.00
	$RMMHC^c\_Single\_GS$	**0.50** $\pm$ 0.00	0.50 $\pm$ 0.00	**1.00** $\pm$ 0.00	**1.0** $\pm$ 0.00
	$RMMHC\_Multiple\_GS$	**0.50** $\pm$ 0.00	**0.65** $\pm$ 0.00	**1.00** $\pm$ 0.00	**1.00** $\pm$ 0.00
	$RMMHC^c\_Multiple\_GS$	**0.50** $\pm$ 0.00	**0.65** $\pm$ 0.00	**1.00** $\pm$ 0.00	**1.00** $\pm$ 0.00
AVG	RGS	0.36	0.39	0.54	0.64
	$RMMHC\_Single\_GS$	0.36	0.48	**0.59**	**0.71**
	$RMMHC^c\_Single\_GS$	0.35	0.48	**0.59**	**0.71**
	$RMMHC\_Multiple\_GS$	0.36	**0.49**	**0.59**	**0.71**
	$RMMHC^c\_Multiple\_GS$	**0.37**	**0.49**	**0.59**	**0.71**

Table 7.13: Experimental protocol N°2 Average $\pm$ standard deviation of s-Recall for each algorithm for a particular sample size and network. Bold values present the best values for a given model and a given sample size. The AVG values present the average s-Recall values for all models for a given sample size.

Model	Algorithm	Data size			
		500	1000	2000	3000
RBN_1	RGS	0.16 ± 0.05	**0.16** ± 0.05	0.25 ± 0.12	**0.70** ± 0.02
	$RMMHC\_Single\_GS$	**0.17** ± 0.05	**0.16** ± 0.05	0.26 ± 0.12	**0.70** ± 0.02
	$RMMHC^c\_Single\_GS$	**0.17** ± 0.05	**0.16** ± 0.05	**0.27** ± 0.12	**0.70** ± 0.02
	$RMMHC\_Multiple\_GS$	**0.17** ± 0.05	**0.16** ± 0.05	0.26 ± 0.12	**0.70** ± 0.02
	$RMMHC^c\_Multiple\_GS$	**0.17** ± 0.05	**0.16** ± 0.05	**0.27** ± 0.12	**0.70** ± 0.02
RBN_2	RGS	0.58 ± 0.08	0.65 ± 0.03	0.75 ± 0.00	0.75 ± 0.00
	$RMMHC\_Single\_GS$	0.55 ± 0.07	**0.80** ± 0.00	**0.89** ± 0.00	**0.89** ± 0.00
	$RMMHC^c\_Single\_GS$	0.55 ± 0.07	**0.80** ± 0.00	**0.89** ± 0.00	**0.89** ± 0.00
	$RMMHC\_Multiple\_GS$	**0.61** ± 0.05	**0.80** ± 0.00	**0.89** ± 0.00	**0.89** ± 0.00
	$RMMHC^c\_Multiple\_GS$	**0.61** ± 0.05	**0.80** ± 0.00	**0.89** ± 0.00	**0.89** ± 0.00
RBN_3	RGS	0.33 ± 0.05	**0.38** ± 0.08	0.41 ± 0.06	0.51 ± 0.06
	$RMMHC\_Single\_GS$	**0.38** ± 0.06	**0.38** ± 0.06	**0.45** ± 0.03	**0.56** ± 0.06
	$RMMHC^c\_Single\_GS$	**0.38** ± 0.06	**0.38** ± 0.06	**0.45** ± 0.03	**0.56** ± 0.06
	$RMMHC\_Multiple\_GS$	0.35 ± 0.07	**0.38** ± 0.06	**0.45** ± 0.03	**0.56** ± 0.06
	$RMMHC^c\_Multiple\_GS$	**0.38** ± 0.06	**0.38** ± 0.06	**0.45** ± 0.03	**0.56** ± 0.06
RBN_4	RGS	**0.25** ± 0.12	0.26 ± 0.10	0.29 ± 0.13	0.29 ± 0.07
	$RMMHC\_Single\_GS$	0.24 ± 0.14	**0.46** ± 0.09	**0.30** ± 0.07	**0.30** ± 0.04
	$RMMHC^c\_Single\_GS$	0.24 ± 0.14	**0.46** ± 0.09	**0.30** ± 0.07	**0.30** ± 0.04
	$RMMHC\_Multiple\_GS$	0.23 ± 0.14	0.40 ± 0.06	**0.30** ± 0.07	**0.30** ± 0.04
	$RMMHC^c\_Multiple\_GS$	0.23 ± 0.14	0.40 ± 0.06	**0.30** ± 0.07	**0.30** ± 0.04
RBN_5	RGS	0.40 ± 0.09	0.38 ± 0.10	0.74 ± 0.07	0.74 ± 0.07
	$RMMHC\_Single\_GS$	0.55 ± 0.04	0.42 ± 0.08	**0.83** ± 0.02	**0.77** ± 0.03
	$RMMHC^c\_Single\_GS$	**0.57** ± 0.03	0.42 ± 0.08	**0.83** ± 0.02	**0.77** ± 0.03
	$RMMHC\_Multiple\_GS$	0.54 ± 0.05	**0.60** ± 0.05	**0.83** ± 0.02	**0.77** ± 0.03
	$RMMHC^c\_Multiple\_GS$	0.55 ± 0.05	**0.60** ± 0.05	**0.83** ± 0.02	**0.77** ± 0.03
AVG	RGS	0.34	0.37	0.50	0.61
	$RMMHC\_Single\_GS$	0.38	0.44	**0.55**	**0.64**
	$RMMHC^c\_Single\_GS$	**0.40**	0.44	**0.55**	**0.64**
	$RMMHC\_Multiple\_GS$	0.38	**0.47**	**0.55**	**0.64**
	$RMMHC^c\_Multiple\_GS$	0.39	**0.47**	**0.55**	**0.64**

Table 7.14: Experimental protocol N°2 Average $\pm$ standard deviation of s-F-Measure for each algorithm for a particular sample size and network. Bold values present the best values for a given model and a given sample size. The AVG values present the average s-F-Measure values for all models for a given sample size.



(a) Average h-Precision (b) Average h-Recall (c) Average h-FMeasure

Figure 7.6: The average values of h-Precision, h-Recall and h-FMeasure with respect to the sample size

$RMMHC\_Multiple\_GS$ especially for small dataset sizes (e.g., RBN_3 and RBN_5 with dataset size equal to 500). Also, $RMMHC^c\_Single\_GS$ provides better results than $RMMHC\_Single\_GS$. Hard and soft precision, recall and f-score have the same values for RBN_1, RBN_2, RBN_3 as $K_{max} \in \{0, 1\}$ for these networks. For RBN_4 their values remain the same also. For RBN_5, we notice a slight difference between hard and soft precision, recall and f-Measure using $RMMHC\_Multiple\_GS$, $RMMHC^c\_Multiple\_GS$ with sample size 1000.

Figure 7.6 illustrate the average values of h-Precision, h-Recall and h-FMeasure with respect to the sample size. We have not illustrates soft values as the since there is too little differences between the reported values.

Quality of reconstruction conclusions. $RMMHC$ algorithm, with its different versions, presents better results than RGS. Figure 7.6(c) provides a weighted average of both precision and recall. Based on h-FMeasure, $RMMHC^c\_Single\_GS$ presents better results than all other $RMMHC$ versions.

7.4 Discussion

7.4.1 Benchmarks and datasets

Tsamardinos et al. (Tsamardinos et al., 2006) have experimentally proved that MMHC for Bayesian networks is able to scale up to thousands of variables in reasonable time while preserving a good learning quality. Unfortunately, currently, we are unable to verify this property for our relational case. Our experimentations were limited to simple models with few number of variables and this is due to some problems raised during benchmark generation.

In fact, we noted that the more the RBN generated is complex and/or the size of the dataset to be sampled is large, the more the ground Bayesian network generation and sampling steps of the benchmark generation process are slow. Knowing that the time complexity of the sampling process is linear on the number of variables, we can conclude that the problem is in the spatial complexity of the generated GBN object from which we are sampling datasets. This problem which presented a handicap for our experimental process led us to seek possible solutions. One possible direction is to process sampling in lifted models rather than grounded ones (cf. Section 3.2.5).

7.4.2 Canonical dependencies generation

We used the concept of canonical order dependencies proposed a relational distance-based. RSHD compares the list of canonical dependencies derived from a learning algorithm to the list of perfect canonical

dependencies derived from the true model. Following the first experimental protocol (cf. Section 7.2.1), we have concluded that such a metric is more suitable to judge the quality of reconstruction of the graph. Yet this solution was not generic enough to work with all DAGs. In addition, we have seen that the construction of this list of canonical dependencies depends on the K_{max} value. The more K_{max} is high, the more the construction of the canonical dependencies list takes time, which makes its use impossible, especially for experimentations where we have hundreds of networks to transform and to compare.

In fact, to provide more rigorous evaluation metrics, we are in need of a graphical representation similar to the essential graph representation (cf. Section 1.3.3) at the relational context. A graphical representation that is derived directly from the RBN dependency structure, just like the CPDAG derived directly from the BN graphical component.

7.4.3 Conservative vs non conservative algorithms

For both experimental protocols we have seen that conservative algorithms provide better results than non conservative one in term of quality of reconstruction. Yet, they made more statistical calls. It would be more interesting to use conservative algorithms if we are working with small data sets.

7.4.4 Learned dependency structure complexity

As described in Section 3.3.1, The RBD score includes a prior that penalizes long indirect slot chains. Yet it does not control the learned dependency structure complexity which may led to some false positive dependencies and affect the quality of the learning approach. To provide further control of the dependency structure complexity we can introduce a new penalty term that penalizes models with many parameters (i.e., complex structures containing a lot of probabilistic dependencies). For standard Bayesian networks, measurement from information theory have been used to address this issue such as AIC (Akaike, 1970) and BIC (Schwarz, 1978) criteria.

7.4.5 Query performance

Also, we noted that if the dataset sizes gets bigger, query performance decreases. Consequently, to be able to run the learning algorithms using large datasets with huge number of variables, we are in need of further solutions in order to speed up query performance. Possible solutions to address this issue would be column-stores or graph databases. DBMS using column store architecture enable queries to read just the attributes they need, rather than having to read entire rows from disk and discard unneeded attributes once they are in memory (Abadi et al., 2013). Graph database are characterized by a relatively constant performance, even with large datasets. This is due to the fact that the execution time for each query is proportional only to the size of the part of the graph traversal to satisfy that query (Robinson et al., 2013).

7.5 Conclusion

Performance evaluation is an essential component in the development process of any data mining tool. The lack of famous benchmarks in our relational data mining context presents a basic limitation to study and compare the proposed approaches. Via this chapter, we showed the utility of our benchmark random generation tool and its major role when evaluating the quality of any RBN structure learning algorithm. The experimental protocols have been made using several generated theoretical RBNs and various relational observational database instances, of different sizes, sampled from those RBNs. Achieving these results was not possible without our benchmark generation process.

In this chapter we have made a first comparative study including all state-of-art structure learning algorithms for relational domain. Thanks to this experimental prototype, we have been able to discuss some crucial tasks that have to be either designed or improved in order to ensure more progress in this area.

We have applied our structure learning evaluation metrics to evaluate the results of the learning approaches included in the comparative study. For both experimental protocols, RMMHC outperforms RGS either in term of complexity (number of statistical calls) or in term of accuracy (precision, recall, f-score and RSHD). In the experimental protocol N°1, we have generated a limited range of models, that RCD is able to learn (cf. Section 7.2.1), and we have compared RMHHC to both RGS and RCD. The results showed that RCD and RMMHC present similar results in term of accuracy. In order to highlight RMMHC learning abilities, we have performed the experimental protocol N°2. For this latter we have excluded RCD from the comparative study as it cannot be performed and we have generated a wide range of models that RMMHC is able to learn (cf. Section 7.2.2).

Conclusion

Summary

The first part of this dissertation was dedicated to a survey on Bayesian networks, relational databases and relational Bayesian networks. Chapter 1 deals with Bayesian networks, their definition, utility and learning approaches allowing their construction from observational data. The chapter also discussed methods for generating and sampling Bayesian networks and techniques to measure the quality of the learning algorithms. Chapter 2 applies to database theory. It provides a quick overview on database management and focuses mainly on the relational representation of the data and relational benchmarking. Chapter 3 concerns relational Bayesian networks, which are based on Bayesian networks and relational data representation discussed in the first two chapters. The chapter provides RBNs definition and existing approaches to learn them from relational data. We have shown that only few methods have been proposed to deal with this issue. Also we have discussed the lack of RBNs generating methods and techniques to measure the quality of an algorithm. These observations were the subject of the second part of this thesis, which is dedicated to the contributions.

Our contributions were developed theoretically in Chapters 4 and 5. Then they were approved experimentally in Chapter 7. Chapter 6 has been dedicated to introduce the PILGRIM project and a major part of the chapter discussed strategies used to implement our contributions. Chapter 4 provides an algorithmic approach allowing to generate random RBNs from scratch to cover the absence of generation process. The proposed method allows to generate RBNs as well as synthetic relational data from a randomly generated relational schema and a random set of probabilistic dependencies. It allows to generate various relational schemas, from simple to complex ones, and to populate database tables with huge number of tuples derived from distributions defined by the generated RBNs. A second part of the chapter concerns the evaluation metrics, where we have redefined the Precision and Recall metrics for the relational context and we have proposed a relational extension of the SHD measure based on the use of canonical dependencies. Chapter 5 proposes a novel approach to learn RBNs structure called Relational Max-Min Hill Climbing algorithm, RMMHC for short. It presents a first hybrid approach proposed to learn RBNs structure from relational observational data. As discussed in Chapters 1 and 3, RBNs are an extension of BNs in the relational context. On the other hand, Hybrid methods to learn BNs structures gave the best results, compared with score-based and constraint-based methods used separately. However, no hybrid approaches have been proposed for RBNs structure learning. The RMMHC algorithm is based on a first local search step performed using the relational max-min Parents an children (RMMPC) algorithm detailed in Chapter 5 and a global search step performed on the basis of the RMMPC results. We have proposed two possible versions of the overall algorithm and we have discussed differences between both.

Chapter 7 approves the proposed learning algorithm with experimental results. It provides a first comparative study of state-of-the-art relational structure learning approaches using standardized evaluation methods. Usually, the evaluation of the learning approaches is generally done using randomly generated data coming from either real known networks or randomly generated ones. However, neither the first nor the second are available as we have discussed in Chapter 3. Consequently we have used the approach described in Chapter 4 to randomly generate RBNs and relational observational data from these generated networks to develop the experimentation part. Algorithms evaluation has been made using the relational

metrics proposed in Chapter 4. We have presented two experimental protocols. The experimental protocol N°1 deals with a limited range of generated models but allows to compare all state-of-the-art algorithms. The experimental protocol N°2 deals with a wide range of generated models that $RMMHC$ is able to learn but excludes the RCD algorithm as it cannot be performed with this range of models. Experimental results showed that our approach presents good results either in term of complexity or in term of accuracy.

Application: a preliminary work

We are about using RBNs to address the recommendation task (cf. Annexe A). In (Ben Ishak et al., 2013) we have proposed a first contribution in this context. Our approach limits the search procedure to a generation of a simple model in order to resolve the model generation computational task noted in (Huang et al., 2005). But this model is not well developed to ensure a good recommendation. Thus, the key component to enrich the model is to provide an appropriate model instantiation to each active user and to perform recommendation at this level. The model instantiation is then performed based on a set of rules derived from recommendation requirements. The advantage of this approach is that it is based on the RBN model and its instantiation and so all the recommendation process is crossed through this model without combining it with other frameworks. In addition, the search procedure is optimized by limiting it to only browse short paths.

Future work

The work presented in this thesis is just the beginning for several challenging research tasks. Since RBNs bring together two neighboring subfields of computer science, namely machine learning and database management, our process can be of interest not only for machine learning researchers but also for database designers to evaluate the effectiveness of a database management system (DBMS) components. It allows to generate various relational schemas, from simple to complex ones, and to populate database tables with huge number of tuples derived from distributions defined by the generated RBNs. The generation process can be improved to give more flexibility in the choice of class attributes, domains, etc. and also to tackle some other features of the relational model: NULL values, composite key (key made of several columns), UNIQUE constraints, etc. Additionally, to be used as a benchmark, it has to be enriched by a test queries component, in order to exhibit particular specificities of the query language, or particular behaviors of database engines.

In Chapter 4, we have presented a relational extension of the SHD measure using canonical dependencies (cf. Section 3.3.2). The application of this measure was limited to a special relational representation (cf. Section 3.2.3). As a future work we aim to provide a generic version of this measure and this is by establishing the notion of equivalent RBN dependency structures. Our goal is to define Markov equivalence class for RBNs in theory and to provide an algorithmic method allowing its construction given a RBN dependency structure.

We have made our generated PRMs as well as data available to researchers who are working in this area [1]. As we lack of PRM benchmarks, we believe that this can be a useful tool for evaluating new proposals related to PRMs learning from relational data. We are also about to distribute our software into GPL license.

Also we aim to apply structure learning approaches to the recommendation task. In (Ben Ishak et al., 2013) we have proposed an architecture of a recommender system based on the use of RBNs. As perspective we aim to implement our proposal and to test it with real recommendation benchmarks.

As discussed in Section 3.2.6, RBNs are characterized by their ability to model more complex structural uncertainty. Consequently, extending our generation and learning approaches to deal with these extensions

1. https://drive.google.com/folderview?id=0B160ZHpTs0CfUGI0Tmc3VXdJX1U&usp=sharing

might be a challenging research area.

Another interesting task is to expand our research to address other directed probabilistic relational models. Learning DAPER (cf. Section 3.2.3) structure using schema-free graph databases is one of our perspectives. The challenge here is to learn both the schema and the dependency structure.

Annex 1: Recommender Systems

RBNs were succesfully applied in several areas such as industry (Medina-Oliva et al., 2010), system quality analysis (Medina-Oliva et al., 2009), web page classification (Fersini et al., 2009), risk analysis (Sommestad et al., 2010), recommendation (Getoor and Saham, 1999) etc. which reflect great interest to upgrade from propositional models to relational ones as they are more convenient and appropriate to the requirements of real-world problems. In what follows we focus on recommendation.

A.1 Recommendation techniques and main issues

Recommender systems (RS) (Ricci et al., 2010; Jannach et al., 2010) emerged in the mid-1990s as a new research area whose interest has increased recently with the intension of reducing part of the information overload problem produced on the Net. They are invoked in many Internet sites such as Amazon, YouTube, Yahoo, Netflix, etc. In 2009, Netflix awarded a million dollar prize to the team that first succeeded in improving substantially the accuracy of predictions of its recommender system [1]. The ultimate goal of a recommender system is to deliver a list of personalized recommended items to a particular user within a specific domain.

Several recommender systems have been developed. Nonetheless, collaborative filtering and content-based approaches stay the most familiar and mature ones (Park et al., 2012). The former attempts to identify groups of users with similar tests as the active user and recommends items that they have liked. The latter learns to recommend items that are similar to those the user has liked in the past. Data mining techniques have been largely applied for the first as for the second approach.

Content-based recommendation approaches analyze features of items previously rated by a user in order to build a profile of user interests. Then, the recommendation process consists in matching up the attributes of the user profile against the attributes of a content item with the intension of providing the user's level of interest in that item.

While content-based recommender systems need only ratings provided by the active user to build his own profile, collaborative filtering methods need ratings from other users in order to find users that have similar tastes since they rated the same items similarly. Then, only the items that are most liked by this group will be recommended.

Each of these approaches presents some deficiencies which present challenging issues when conceiving a recommendation approach. Among them we can list:

1. http://www.netflixprize.com/

Data sparsity. In general users rate only a limited number of items, whereas recommender systems are based on large datasets. Consequently, the user-item matrix could be extremely large and sparse, which affects the performances of the recommendation. One direct consequence of data sparsity is the cold start problem.

Cold start problem. Cold start refers to the difficulty in providing recommendation for new users or new items due to the lack of information: To provide reliable recommendations, the system needs the user ratings, this is not supplied with a new user. Likewise, new items need to be rated before they could be recommended. This problem does not arise for the content-based technique as the recommendation of an item is based on its characteristics rather than its ratings. While content-based approaches have also a start-up problem in that they must build a reliable classifier about user with very few ratings.

Scalability. Often high scalability of recommender systems is needed, especially that many systems need to react immediately to online requirements. Knowing that they operate with millions of users and million of items, trade-offs between scalability and prediction performance have to be established.

There have been studies to integrate both content-based and collaborative filtering strategies into hybrid systems in order to complement each other's deficiency (Burke, 2002). On the other hand, several data and knowledge sources can be available for a recommender system, however, their exploitation depends on the used recommendation technique. Various recommendation approaches are derived from machine learning techniques, commonly classification and clustering methods (de Campos et al., 2010; Xu et al., 2012), and rely on simple data representation of the user-item rating matrix. Some avenues of research attempt to use additional domain knowledge to the classical user-item interaction with the intension to achieve better performance (Carrer-Neto et al., 2012; Deng et al., 2011; Kapusuzoglu and Öguducu, 2011).

In the following we will be interested in a set of approaches that uses RBNs to perform recommendation. We will show different manner of the integration of such a model to the recommendation process. Also we will show that the use of RBNs allows to build an hybrid system due to the integration of all the available information.

A.2 RBNs for recommendation

A.2.1 Reviews

The first proposition to apply RBN to recommendation was proposed by (Getoor and Saham, 1999) Their approach is based on the models proposed by (Ungar and Foster, 1998; Hofmann and Puzicha, 1999) which use BNs for collaborative filtering. The main idea in these models is to cluster the users and the items separately and make prediction based on the clusters instead of working on a large set of user-item matrix. In this model, strong assumptions are made that each person (also each item) belongs to only one cluster and that every relation r_{ij} must have the same local probability model. These assumptions make it possible to represent this model compactly by RBNs.

Newton (Newton and Greiner, 2004) has proposed a new way of hierarchal RBN: They organize items to hierarchal types tree, next create a class type for every leaf node in the tree and then divide the classes reachable from classes' types previously created. Hierarchal for a class X will divide the classes that are reachable from the class X for one or more reference slot. For example if we have a system in which users give score to movies and we have four different types of movies, so we will have four different scores. In this model the hierarchy is not well exploited as only leaves are represented in the hRBN.

Gao et al. (Gao et al., 2007) described a method that combines Collaborative Filtering (CF) and RBNs. This method develops a User grade (UG) function to determine the weight between the prediction from CF and that from the RBN. The RBN considered in the model employs user demographic information and movie's genre, but its structure is very simple and slot chains' length are no greater than one; hence, it does

not really exploit the properties of RBNs. Finally, it must be noticed that, as the UG function increases, UGCF-RBN is dominated by CF and RBN is relegated to solve the problems of regular CF.

Huang et al. (Huang et al., 2005) proposed a unified approach based on RBNs structure learning. In fact in their work they develop an hybrid approach able to combine different techniques of recommendation and thus different informations gathered by the RS (i.e., demographic information about users, items characteristics, transactions,...) and by this way they can overcome shortcomings of each of these techniques and also provide promising results in term of accuracy and performance comparing to existing approaches. The idea besides their proposal is to walk through long slot chains in order to capture and represent the maximum of information needed for recommendation: For instance, in CF techniques, we need to represent neighbors properties this is cannot be done using simple slots. They have shown that the most long slot chains are considered the most the model will capture interesting patterns for recommendation but also the most the RBN model will be complex and the process of its generation will be a computational task. In order to optimize their approach, they only use the greedy search algorithm for RBN structure learning (Friedman et al., 1999a) from a recommendation perspective where they focus only on finding dependencies of the exist node from the order class with other classes' attributes in order to find nodes constructing its Markov blanket. Then, they use this attributes to build a naive classifier and to perform recommendation using probabilistic inference on this net.

Recently, (Chulyadyo and Leray, 2014) proposed a personalized recommender system based on RBN with existence uncertainty. The specificity of this proposal is that it allows to build a personalized recommender in cold start situation, when no user profile exists.

A.2.2 Discussion

(Getoor and Saham, 1999)'s work builds a hybrid recommender system rather than a collaborative one as explained in the paper. The good points of this model are that it is easier to interpret than a Bayesian network, is extensible and can handle cold start and scalability problems. However, the paper does not explain how to learn such RBN and how to make actual recommendations from such model. Besides, the effectiveness of the model mainly depends on the quality of clusters and it is difficult to obtain good clusters that can assign items/users to only one cluster. Also, the authors have not presented experimental results. Hence, we do not have a clear vision of how effective the model could be.

The introduction of mutli-set operation has increased the expressiveness of the recommendation framework proposed by (Huang et al., 2005). It allows to capture data patterns that cannot be captured by simple (short) slot chains of RBNs. The framework unifies different recommendation techniques simultaneously. They have presented promising experimental results whereas the obvious limitation of this work persists on the intensive computation required by the model estimation process. In addition, this framework aims at dealing with binary transaction data rather than rating data. Customers' purchase history is an example of binary transaction data where the recommendation problem is to predict the existence of a customer-product pair. So, a new attribute *Exists* is introduced in the relational data to indicate whether the pair exists. All the records in purchase history will have $Exists = 1$ and for the customer-product pairs that are not in the purchase history take $Exists = 0$. The objective is to create a classifier with $Exists$ attribute as a target attribute and predict the probability of the existence of the customer-product pair given other attributes.

(Newton and Greiner, 2004)'s work addresses the cold start problem as for new users the scoring will depend on the demography of the user and for the new items it depends on the demography of the user and the users previous scoring for other movies types. In addition, Different types of scores can depend on different attributes. They show that hierarchal RBNs perform better than standard RBNs and some other algorithms however, the approach presents some limitations. for instance, for a given person X, his score cannot depend on his previous score of the same type. Also Hierarchical classes are not well represented, as only leaves are represented in the hierarchal RBN. In which it could be considered more as a clustering instead.

(Gao et al., 2007)'s work describes a method that combines Collaborative Filtering (CF) and RBNs

which aims to overcome simultaneously the most common problems of CF: sparsity, scalability and cold start. They showed that these issues are solved by the use of RBNs. However this work does not exploit the capability of RBN in recommendation. For the target user with many ratings, the effect of RBN becomes negligible.

(Chulyadyo and Leray, 2014)'s work is a pure RBN-based approach and allows to perform recommendations in both cold and hot systems using the same RBN model. Exprimental results were interesting, yet they applied an offline evaluation approach.

Bibliography

Abadi, D., Boncz, P., Harizopoulos, S., Idreos, S., and Madden, S. (2013). The design and implementation of modern column-oriented database systems. *Foundations and Trends in Databases*, 5:197–280. 133

Acid, S. and de Campos, L. M. (1996). Benedict: an algorithm for learning probabilistic Bayesian networks. In *Proceedings of the Sixth International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems*, pages 979–984. 18

Acid, S. and de Campos, L. M. (2000). Learning right sized belief networks by means of a hybrid methodology. In *PKDD*, Lecture Notes in Computer Science, pages 309–315. Springer. 18

Acid, S. and de Campos, L. M. (2001). A hybrid methodology for learning belief networks: Benedict. *International Journal of Approximate Reasoning*, 27:235–262. 18

Acid, S. and de Campos, L. M. (2003). Searching for Bayesian network structures in the space of restricted acyclic partially directed graphs. *Journal of Artificial Intelligence Research*, 18:445–490. 19

Akaike, H. (1970). Statistical predictor identification. *Ann. Inst. Statist. Math.*, 22:203–217. 17, 133

Andersson, S., Madigan, D., and Perlman, M. D. (1997). A characterization of Markov equivalence classes for acyclic digraphs. *Annals of Statistics*, 25(2):505–541. 14

Andreassen, S., Jensen, F. V., Andersen, S. K., Falck, B., Kjaerulff, U., Woldbye, M., Sørensen, A. R., Rosenfalck, A., and Jensen, F. (1989). *MUNIN - an Expert EMG Assistant. In Computer-Aided Electromyography and Expert Systems, Chapter 21*. Elsevier, Noth-Holland. 22

Angles, R., Prat-Pérez, A., Dominguez-Sal, D., and Larriba-Pey, J.-L. (2013). Benchmarking database systems for social network applications. In *First International Workshop on Graph Data Management Experiences and Systems*, GRADES '13, pages 1–7, New York, NY, USA. ACM. 38

Arias, M., Díez, F. J., and Palacios, M. P. (2011). ProbModelXML. A format for encoding probabilistic graphical models. Technical Report CISIAD-11-02, UNED, Madrid, Spain. 105

Bachman, C. W. (1969). Data structure diagrams. *SIGMIS Database*, 1(2):4–10. 36

Ballinger, C. (1993). Tpc-d: Benchmarking for decision support. In Gray, J., editor, *The Benchmark Handbook*. Morgan Kaufmann. 39

Bangsø, O. and Wuillemin, P.-H. (2000). Object Oriented Bayesian networks: a framework for top-down specification of large Bayesian networks with repetitive structures. Technical report, Department of Computer Science, Aalborg University, Denmark. 2

Ben Ishak, M., Ben Amor, N., and Leray, P. (2013). A rbn-based recommender system architecture. In *Proceedings of the 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO 2013)*, pages 1–6, Hammamet, Tunisia. 4, 136

Ben Ishak, M., Chulyado, R., Abdelwahab, A., Ramirez, M., Leray, P., and Ben Amor, N. (2014a). Relational Bayesian networks for recommender systems: review and comparative study. The 2014 ENBIS-SFdS Spring Meeting on graphical causality models. Paris, France. 4

Ben Ishak, M., Leray, P., and Ben Amor, N. Probabilistic relational model benchmark generation: Principle and application. *Intelligent Data Analysis International Journal (to appear)*, pages ?–? 4

Ben Ishak, M., Leray, P., and Ben Amor, N. (2011a). Ontology-based generation of Object Oriented Bayesian Networks. In *Proceedings of the 8th UAI Bayesian Modeling Applications Workshop (UAI-AW 2011)*, pages 9–17, Barcelona, Spain. 4

Ben Ishak, M., Leray, P., and Ben Amor, N. (2011b). A two-way approach for probabilistic graphical models structure learning and ontology enrichment. In *Proceedings of the 3rd International Conference on Knowledge Engineering and Ontology Development (KEOD 2011) part of the International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management IC3K*, pages 189–194, Paris, France. 4

Ben Ishak, M., Leray, P., and Ben Amor, N. (2014b). La génération aléatoire de réseaux bayésiens relationnels. 7èmes journées francophones de réseaux Bayésiens. Paris, France. 4

Ben Ishak, M., Leray, P., and Ben Amor, N. (2014c). Random generation and population of probabilistic relational models and databases. In *Proceedings of the 26th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2014)*, pages 756–763. 4

Bender, E. A. and Robinson, R. W. (1988). The asymptotic number of acyclic digraphs, II. *J. Comb. Theory, Ser. B*, 44(3):363–369. 74

Binder, J., Koller, D., Russell, S., and Kanazawa, K. (1997). Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29(2–3):213–244. 22

Bitton, D., Turbyfill, C., and Dewitt, D. J. (1983). Benchmarking database systems: A systematic approach. In *Proceedings of the 9th International Conference on Very Large Data Bases*, pages 8–19. ACM. 39

Bouckaert, R. R. (1993). Probalistic network construction using the minimum description length principle. In *ECSQARU'93*, volume 747 of *Lecture Notes in Computer Science*, pages 41–48. Springer. 17

Bruno, N. and Chaudhuri, S. (2005). Flexible database generators. In *Proceedings of the 31st International Conference on Very Large Data Bases*, pages 1097–1107. ACM. 39

Buntine, W. (1991). Theory refinement on Bayesian networks. In *Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence*, pages 52–60. 18

Buntine, W. L. (1994). Operations for learning with graphical models. *Journal of Artificial Intelligence Research*, 2:159–225. 2

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370. 140

Carrer-Neto, W., Hernández-Alcaraz, M., Valencia-García, R., and García-Sánchez, F. (2012). Social knowledge-based recommender system. application to the movies domain. *Expert Systems with applications*, 39(12):10990–11000. 140

Cattell, R. G. G. (1993). The engineering database benchmark. In Gray, J., editor, *The Benchmark Handbook*. Morgan Kaufmann. 38

Chen, P. (1976). The entity-relationship model—toward a unified view of data. *ACM Trans. Database Syst.*, 1(1):9–36. 36

Chickering, D. M. (2002). Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554. xvii, 14, 15, 17, 18

Chickering, D. M., Geiger, D., and Heckerman, D. (1994). Learning Bayesian networks is NP-hard. Technical report, MSR-TR-94-17, Microsoft Research. 16

Chickering, D. M. and Maxwell, D. (2002). Learning equivalence classes of Bayesian-network structures. *J. Mach. Learn. Res.*, 2:445–498. 17

Chow, C. and Liu, C. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Trans. Inf. Theor.*, 14(3):462–467. 17

Chulyadyo, R. and Leray, P. (2014). A personalized recommender system from probabilistic relational model and users' preferences. In *Proceedings of the 18th Annual Conference on Knowledge-Based and Intelligent Information & Engineering Systems*, pages 1063–1072. 141, 142

Codd, E. (1983). A relational model of data for large shared data banks. *Commun. ACM*, 26:64–69. 29

Cooper, G. (1990). Computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42:393–405. 15

Cooper, G. and Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347. 17

Curino, C. A., Difallah, D. E., Pavlo, A., and Cudre-Mauroux, P. (2012). Benchmarking OLTP/Web Databases in the Cloud: The OLTP-bench framework. In *Proceedings of the Fourth International Workshop on Cloud Data Management*, CloudDB '12, pages 17–20, New York, NY, USA. ACM. 38

Daly, R., Shen, Q., and Aitken, S. (2011). Learning Bayesian networks: approaches and issues. *The Knowledge Engineering Review*, 26:99–157. 16

Darmont, J. (2009). Database benchmarks. In Erickson, J., editor, *Database Technologies: Concepts, Methodologies, Tools, and Applications*, pages 1226–1233. IGI Global. 40

Darwiche, A. (2009). *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, New York, NY, USA, 1st edition. 16

Dash, D. and Druzdzel, M. J. (1999). A hybrid anytime algorithm for the construction of causal models from sparse data. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 142–149. 19

Date, C. (2003). *An Introduction to Database Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 8 edition. 31, 32, 36

Date, C. J. (1990). *Relational database writings: 1985-1989*. Addison-Wesley. 36, 38

Date, C. J. (2005). *Database in Depth: Relational Theory for Practitioners: The Relational Model for Practitioners*. O'Reilly Media, 1 edition. 31

Date, C. J. (2008). *The Relational Database Dictionary, Extended Edition*. Apress, New York. 31, 34

de Morais, S. R. and Aussem, A. (2010). A novel markov boundary based feature subset selection algorithm. *Neurocomputing*, 73(4–6):578–584. 21, 90

de Campos, L. M. (2006). A scoring function for learning Bayesian networks based on mutual information and conditional independence tests. *Journal of Machine Learning Research*, 7:2149–2187. 25

de Campos, L. M., Fernández-Luna, J., Huete, J., and Rueda-Morales, M. (2010). Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks. *Int. J. Approx. Reasoning*, 51(7):785–799. 140

de Jongh, M. and Druzdzel, M. J. (2014). Evaluation of rules for coping with insufficient data in constraint-based search algorithms. In *Proceedings of the 7th Probabilistic Graphical Models*, pages 190–205. 21

De Raedt, L. (1998). Attribute-value learning versus inductive logic programming: the missing links. In *Proceedings of the Eighth International Conference on Inductive Logic Programming*, pages 1–8. 1

Deng, Y., Wu, Z., Si, H., Xiong, H., and Chen, Z. (2011). A collaborative filtering approach to making recommendations based on ontology in the movie domain. *Energy Procedia*, 13:228–236. 140

DeWitt, D. J. (1993). The wisconsin benchmark: Past, present, and future. In Gray, J., editor, *The Benchmark Handbook*. Morgan Kaufmann. 38

Difallah, D. E., Pavlo, A., Curino, C., and Cudre-Mauroux, P. (2013). OLTP-Bench: An extensible testbed for benchmarking relational databases. 7(4):277–288. 38

FAST, A. S. (2010). *Learning the structure of Bayesian networks with constraint satisfaction*. PhD thesis, University of Massachusetts Amherst. 25

Ferrarons, J., Adhana, M., Colmenares, C., Pietrowska, S., Bentayeb, F., and Darmont, J. (2013). Primeball: a parallel processing framework benchmark for big data applications in the cloud. In *5th TPC Technology Conference on Performance Evaluation and Benchmarking (VLDB/TPCTC 13), Riva del Garda, Italy*, Lecture Notes in Computer Science, pages 109–124, Heidelberg, Germany. Springer. 40

Fersini, E., Messina, E., and Archetti, F. (2009). Probabilistic relational models with relational uncertainty: An early study in web page classification. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 03*, WI-IAT '09, pages 139–142, Washington, DC, USA. IEEE Computer Society. 41, 139

Friedman, N., Getoor, L., Koller, D., and Pfeffer, A. (1999a). Learning probabilistic relational models. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1300–1309. xvii, 2, 17, 55, 56, 85, 86, 96, 141

Friedman, N., Nachman, I., and Peér, D. (1999b). Learning Bayesian network structure from massive datasets: The sparse candidate algorithm. In *Proceedings of the 15th UAI*, pages 206–215. 19

Fung, R. M. and Chang, K. (1990). Weighing and integrating evidence for stochastic simulation in Bayesian networks. In *Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence*, pages 209–220. 24

Gao, Y., Hong, H., Liu, J., and Liu, D. (2007). A recommendation algorithm combining user grade-based collaborative filtering and probabilistic relational models. In *Proceedings of the Fourth International Conference on Fuzzy Systems and Knowledge Discovery*, pages 67–71, Washington, DC, USA. IEEE Computer Society. 140, 141

Gelfand, A. E. and Smith, A. F. M. (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85:398–409. 24

Getoor, L. (2002). *Learning statistical models from relational data*. PhD thesis, Stanford University. 55, 71, 72, 87

Getoor, L., Koller, D., Friedman, N., Pfeffer, A., and Taskar., B. (2007). *Probabilistic Relational Models, In Getoor, L., and Taskar, B., eds., Introduction to Statistical Relational Learning*. MA: MIT Press, Cambridge. xv, 2, 43, 45, 47, 54, 86, 107, 108

Getoor, L. and Saham, M. (1999). Using probabilistic relational models for collaborative filtering. In *Working Notes of the KDD Workshop on Web Usage Analysis and User Profiling*. 139, 140, 141

Gonzales, C. and Wuillemin, P.-H. (2011). PRM inference using jaffray & faÿ's local conditioning. *Theory and Decision*, 71(1):33–62. 53

Gray, J. (1993). *The Benchmark Handbook: For Database and Transaction Processing Systems*. Morgan Kaufmann, San Francisco, CA, USA. 38, 39

Gray, J., Sundaresan, P., Englert, S., Baclawski, K., and Weinberger, P. J. (1994). Quickly generating billion-record synthetic databases. In *Proceedings of the 1994 ACM SIGMOD international conference on Management of data*, pages 243–252. ACM. 39

Gyftodimos, E. and Flach, P. (2002). Hierarchical Bayesian networks: A probabilistic reasoning model for structured domains. In *Proceedings of the ICML-2002 Workshop on Development of Representations*, pages 23–30, University of New South Wales, Sydney, Australia. 2, 28

Heckerman, D. (1998). A tutorial on learning with Bayesian network. In *Proceedings of the NATO Advanced Study Institute on Learning in graphical models*, pages 301–354, Kluwer Academic Publishers Norwell, MA, USA. xvii, 16, 17, 18, 19, 107, 108

Heckerman, D., Chickering, D. M., Meek, C., Rounthwaite, R., and Kadiel, C. (2001). Dependency Networks for Inference, Collaborative Filtering, and Data Visualization. *Journal of Machine Learning Research*, 1:49–75. 1

Heckerman, D., Geiger, D., and Chickering, D. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243. 18

Heckerman, D., Meek, C., and Koller, D. (2004). Probabilistic models for relational data. Technical report, Microsoft Research, Redmond, WA. xiii, 2, 47, 48, 62

Heckerman, D., Meek, C., and Koller, D. (2007). *Probabilistic entity-relationship models, PRMs, and plate models, In Getoor, L., and Taskar, B., eds., Introduction to Statistical Relational Learning*. MA: MIT Press, Cambridge. 1, 41, 43

Henrion, M. (1986). Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In *Proceedings of Uncertainty in Artificial Intelligence 2 Annual Conference on Uncertainty in Artificial Intelligence (UAI-86)*, pages 149–163, Amsterdam, NL. Elsevier Science. xvii, 24

Hofmann, T. and Puzicha, J. (1999). Latent class models for collaborative filtering. In *International Joint Conference on Artificial Intelligence*, volume 16, pages 688–693. 140

Huang, Z., Zeng, D., and Chen, H. (2005). A unified recommendation framework based on probabilistic relational models. Technical report, The University of Arizona, Tucson. 136, 141

Ide, J. S. and Cozman, F. G. (2002). Random generation of Bayesian networks. In *Brazilian symp.on artificial intelligence*, pages 366–375. Springer-Verlag. 22

Ide, J. S., Cozman, F. G., and Ramos, F. T. (2004). Generating random Bayesian networks with constraints on induced width. In *Proceedings of the 16th Eureopean Conference on Artificial Intelligence*, pages 323–327. xvii, 22, 23, 24

Jaeger, M. (1997). Relational Bayesian networks. In *Proceedings of the 13th Annual Conference on Uncertainty in AI UAI*, pages 266–273. Morgan Kaufmann. 1, 2, 41, 43

Jannach, D., Zanker, M., Felfernig, A., and Friedrich, G. (2010). *Recommender Systems: An Introduction*. Cambridge University Press. 139

Jensen, D. and Neville, J. (2002). Linkage and autocorrelation cause feature selection bias in relational learning. In *Proceedings of the Nineteenth International Conferenceon Machine Learning*, pages 259–266. 59

Jensen, F. V. and Nielsen, T. D. (2007). *Bayesian Networks and Decision Graphs*. Springer, New York, NY, 2nd edition. 16

Kapusuzoglu, H. and Öguducu, S. (2011). A relational recommender system based on domain ontology. In *Proceedings of the 2011 International Conference on Emerging Intelligent Data and Web Technologies*, pages 36–41, Washington, DC, USA. IEEE Computer Society. 140

Kim, J. and Pearl, J. (1983). A computational model for combined causal and diagnostic reasoning in inference systems. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, pages 190–193, Karlsruhe, Germany. 14

Kisynski, J. and Poole, D. (2009). Lifted aggregation in directed first-order probabilistic models. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, pages 1922–1929, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. 53

Kline, K., Klinel, D., and Hunt, B. (2008). *SQL in a nutshell - a desktop quick reference (3. ed.).* O'Reilly. 34

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models*. MIT Press, Cambridge. 2, 7

Koller, D. and Pfeffer, A. (1997). Object-Oriented Bayesian networks. In *Proceedings of the 13th conference on Uncertainty in Artificial Intelligence*, pages 302–313, Providence, Rhode Island, USA. Morgan Kaufmann. 2

Koller, D. and Pfeffer, A. (1998). Probabilistic frame-based systems. In *Proc. AAAI*, pages 580–587. AAAI Press. 1, 2, 28, 41, 43, 47

Kullback, S. and Leibler, R. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 22:79–86. 26

Larrañaga, P., Karshenas, H., Bielza, C., and Santana, R. (2013). A review on evolutionary algorithms in Bayesian network learning and inference tasks. *Inf. Sci.*, 233:109–125. 16

Laskey, K. B. (2008). MEBN: A language for first-order Bayesian knowledge bases. *Artificial Intelligence*, 172:140–178. 2

Lauritzen, S. and Speigelhalter, D. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Royal Statistical Society: Series B (Statistical Methodology)*, 50(2):157–224. 14, 22

Lavrac, N. and Dzeroski, S. (1993). *Inductive Logic Programming: Techniques and Applications*. Routledge, New York. 1

Lu, W., Miklau, G., and Gupta, V. (2014). Generating private synthetic databases for untrusted system evaluation. In *"Proceedings of the 30th IEEE International Conference on data engineering ICDE*, pages 652–663. 39

Maier, M. (2014). *Causal Discovery for Relational Domains: Representation, Reasoning, and Learning*. PhD thesis, University of Massachusetts Amherst. 49

Maier, M., Marazopoulou, K., Arbour, D., and Jensen, D. (2013a). A sound and complete algorithm for learning causal models from relational data. In *Proceedings of the Twenty-ninth Conference on Uncertainty in Artificial Intelligence*, pages 371–380. xv, xvii, 2, 58, 59, 61, 77, 78, 89, 113, 114, 115

Maier, M., Marazopoulou, K., and Jensen, D. (2013b). Reasoning about independence in probabilistic models of relational data. In *Approaches to Causal Structure Learning Workshop, UAI 2013*. xv, 50, 52

Maier, M., Marazopoulou, K., and Jensen, D. (2013c). Reasoning about independence in probabilistic models of relational data. *CoRR*, abs/1302.4381. 47, 49, 50, 51, 53, 58, 61, 95

Maier, M., Taylor, B., Oktay, H., and Jensen, D. (2010). Learning causal models of relational domains. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, pages 531–538. xv, 2, 57, 61, 96

Medina-Oliva, G., Weber, P., Levrat, E., and Iung, B. (2009). Probabilistic relational model (PRM)_based technical knowledge formalization for dependability of an industrial system. In *7th Workshop on Advanced Control and Diagnosis, ACD'2009*, Zielona Gora, Poland. 139

Medina-Oliva, G., Weber, P., Levrat, E., and Iung, B. (2010). Use of probabilistic relational model (PRM) for dependability analysis of complex systems. In *12th IFAC Symposium on Large Scale Systems: Theory and Applications, LSS 2010*, Villeneuve d'Ascq, France. 139

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M., Teller, A., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092. 24

Milch, B., Zettlemoyer, L. S., Kersting, K., Haimes, M., and Pack Kaelbling, L. (2008). Lifted probabilistic inference with counting formulas. In *Proceedings of the 23rd Conference on Artificial Intelligence (AAAI)*. 53

Muggleton, S. and De Raedt, L. (1994). Inductive logic programming: Theory and methods. *JOURNAL OF LOGIC PROGRAMMING*, 19(20):629–679. 1

Murphy, K. P. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, Berkeley, Computer Science Division. 2, 28

Naïm, P., Wuillemin, P.-H., Leray, P., Pourret, O., and Becker, A. (2004). *Réseaux bayésiens*. Eyrolles, Paris. 13, 22

Neville, J. and Jensen, D. (2007). Relational dependency networks. *Journal of Machine Learning Research*, 8:653–692. 1, 41, 62

Neville, J., Jensen, D., Friedland, L., and Hayl, M. (2003a). Learning relational probability trees. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 625–630, New York, NY, USA. ACM. 62

Neville, J., Jensen, D., and Gallagher, B. (2003b). Simple estimators for relational Bayesian classifiers. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), December 19-22, 2003, Melbourne, Florida, USA*, pages 609–612. IEEE Computer Society, Washington, DC, USA. 62

Newton, J. and Greiner, R. (2004). Hierarchical probabilistic relational models for collaborative filtering. In *Proceedings of the 21st International Conference on Machine Learning, Workshop on Statistical Relational Learning*, pages 249–263. 140, 141

Park, D. H., Kim, H. K., Choi, I. Y., and Kim, J. K. (2012). A literature review and classification of recommender systems research. *Expert Syst. Appl.*, 39(11):10059–10072. 139

Pearl, J. (1982). Reverend Bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the AAAI National Conference on AI*, pages 133–136, Pittsburgh, PA. 14

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems*. Morgan Kaufmann, San Franciscos. 1, 2, 7, 9, 12, 14

Pearl, J. (2000). *Causality: Models, reasoning and inference*. MIT Press, Cambridge. 2, 12, 15, 28

Peña, J. M., Nilsson, R., Björkegren, J., and Tegnér, J. (2007). Towards scalable and data efficient learning of markov boundaries. *Int. J. Approx. Reasoning*, 45(2):211–232. 21

Perlich, C. and Provost, F. (2006). Distribution-based aggregation for relational learning with identifier attributes. *Machine Learning*, 62(1-2):65–105. 35

Pfeffer, A. and Koller, D. (2000). Semantics and inference for recursive probability models. In *AAAI/IAAI*, pages 538–544. AAAI Press / The MIT Press. 53

Pfeffer, A. J. (2000). *Probabilistic Reasoning for Complex Systems*. PhD thesis, Stanford University. 1, 2, 41, 43, 45, 47, 53, 54

Provan, G. M. and Singh, M. (1995). Learning Bayesian networks using feature selection. In *D. Fisher and H. Lenz, eds, Proceedings of the fifth International Workshop on Artificial Intelligence and Statistics, Ft. Lauderdale, FL*, pages 450–456. 18

Ricci, F., Rokach, L., Shapira, B., and Kantor, P. (2010). *Recommender Systems Handbook*. Springer, New York, USA. 139

Robinson, I., Webber, J., and Eifreml, E. (2013). *Graph Databases*. O'Reilly Media, Inc. 133

Robinson, R. W. (1977). *Counting unlabeled acyclic digraphs, In C. H. C. LITTLE, Ed., Combinatorial Mathematics V, volume 622 of Lecture Notes in Mathematics*. Springer, Berlin / Heidelberg. 74

Schwarz, G. (1978). Estimating the dimensions of a model. *Annals of Statistics*, 6:461–464. 17, 133

Singh, M. and Valtorta, M. (1993). An algorithm for the construction of Bayesian network structures from data. In *Proceedings ofthe Ninth Conference on Uncertainty in Artificial Intelligence (UAI-93)*, pages 59–265. 18

Singh, M. and Valtorta, M. (1995). Construction of Bayesian network structures from data: a brief survey and an efficient algorithm. *International Journal of Approximate Reasoning*, 12:111–131. 18

Smith, A. F. M. and Roberts, G. (1993). Bayesian computation via the gibbs sampler and related markov chain monte carlo methods (with discussion). *Journal of the Royal Statistical Society*, 55:3–23. 24

Sommestad, T., Ekstedt, M., and Johnson, P. (2010). A probabilistic relational model for security risk analysis. *Computers & Security*, 29:659–679. 41, 139

Spirtes, P., Glymour, C., and Scheines, R. (1990). Causality from probability. In *J. Tiles, G. McKee, and G. Dean (eds.): Evolving Knowledge in the Natural and Behavioral Sciences*, pages 181–199. 16

Spirtes, P., Glymour, C., and Scheines, R. (2000). *Causation, Prediction and Search*. MIT Press. 16, 17, 21

Srebro, N. (2001). Learning markov networks: maximum bounded tree-width graphs. In *Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms*, pages 392–401. 17

Statnikov, A. R., Tsamardinos, I., and Aliferis, C. (2003). An algorithm for generation of large Bayesian networks. Technical report, Department of Biomedical Informatics, Discovery Systems Laboratory, Vanderbilt University. 22

Sumathi, S. and Esakkirajan, S. (2007). *Fundamentals of Relational Database Management Systems*. Springer. 36, 37

Taskar, B., Abbeel, P., and Koller, D. Discriminative probabilistic models for relational data. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*. 1

Torti, L., Wuillemin, P. H., and Gonzales, C. (2010). Reinforcing the Object-Oriented aspect of probabilistic relational models. In *Proceedings of the 5th Probabilistic Graphical Models*, pages 273–280. 61

Tsamardinos, I., Brown, L. E., and Aliferis, C. F. (2006). The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65:31–78. xvii, 2, 19, 20, 21, 25, 26, 83, 85, 88, 132

Ungar, L. and Foster, D. (1998). A formal statistical approach to collaborative filtering. *CONALD'98*. 140

Verma, T. and Pearl, J. (1990). Equivalence and synthesis of causal models. In *Proceedings of the 6th UAI*, pages 220–227. 14, 16

Wellman, M., Breese, J. S., and Goldman, R. P. (1992). From knowledge bases to decision models. *The Knowledge Engineering Review*, 7:35–53. 2

Wuillemini, P. H. and Torti, L. (2012). Structured probabilistic inference. *Int. J. Approx. Reasoning*, 53(7):946–968. 53, 61, 63

Xu, B., Bu, J., Chen, C., and Cai, D. (2012). An exploration of improving collaborative recommender systems via user-item subgroups. In *Proceedings of the 21st international conference on World Wide Web*, pages 21–30. 140

Zhu, Y., Zhan, J., Weng, C., Nambiar, R., Zhang, J., Chen, X., and Wang, L. (2014). Bigop: Generating comprehensive big data workloads as a benchmarking framework. In *Proceedings of DASFAA*, pages 483–492. Springer. 39

l'université nantes angers le mans

PÔLE DE RECHERCHE ET D'ENSEIGNEMENT SUPÉRIEUR

UNIVERSITÉ DE NANTES

Thèse de Doctorat

Mouna BEN ISHAK

Les modèles probabilistes relationnels : apprentissage et évaluation

Cas des réseaux bayésiens relationnels

Probabilistic relational models: learning and evaluation

The relational Bayesian networks case

Résumé

L'apprentissage statistique relationnel est apparu au début des années 2000 comme un nouveau domaine de l'apprentissage machine permettant de raisonner d'une manière efficace et robuste directement sur des structures de données relationnelles. Plusieurs méthodes classiques de fouille de données ont été adaptées pour application directe sur des données relationnelles. Les réseaux Bayésiens Relationnels (RBR) présentent une extension des réseaux Bayésiens (RB) dans ce contexte. Pour se servir de ce modèle, il faut tout d'abord le construire : la structure et les paramètres du RBR doivent être définis à la main ou être appris à partir d'une instance de base de données relationnelle.
L'apprentissage de la structure reste toujours le problème le plus compliqué puisqu'il se situe dans la classe des problèmes NP-difficiles. Les méthodes d'apprentissage de la structure des RBR existantes sont inspirées des méthodes classique de l'apprentissage de la structure des RB. Pour pouvoir juger la qualité d'un algorithme d'apprentissage de la structure d'un RBR, il faut avoir des données de test et des mesures d'évaluation. Pour les RB les données sont souvent issues de benchmarks existants. Sinon, des processus de génération aléatoire du modèle et des données sont mis en oeuvre. Les deux pratiques sont quasi absentes pour les RBR. De plus, les mesures d'évaluation de la qualité d'un algorithme d 'apprentissage de la structure d'un RBR ne sont pas encore établies.
Dans ce travail de thèse, nous proposons deux contributions majeures. I)Une approche de génération de RBR allant de la génération du schéma relationnel, de la structure de dépendance et des tables de probabilités à l'instanciation de ce modèle et la population d'une base de données relationnelle. Nous discutons aussi de l'adaptation des mesures d'évaluation des algorithmes d'apprentissage de RBs dans le contexte relationnel et nous proposons de nouvelles mesures d'évaluation. II) Une approche hybride pour l'apprentissage de la structure des RBR. Cette approche présente une extension de l'algorithme MMHC dans le contexte relationnel. Nous menons une étude expérimentale permettant de comparer ce nouvel algorithme d'apprentissage avec les approches déjà existantes.

Mots clés

Réseaux Bayésiens Relationnels (RBR), Apprentissage de la structure des RBR, Génération de modèles, Mesures d'évaluation.

Abstract

Statistical relational learning (SRL) appeared in the early 2000s as a new field of machine learning that enables effective and robust reasoning about relational data structures. Several conventional data mining methods have been adapted for direct application to relational data representation. Relational Bayesian Networks (RBNs) extend Bayesian networks (BNs) to a relational data mining context. To use this model, it is first necessary to build it: the structure and parameters of a RBN must be set manually or learned from a relational observational dataset. Learning the structure remains the most complicated issue as it is a NP-hard problem. Existing approaches for RBNs structure learning are inspired from classical methods of learning the structure of BNs. The evaluation of learning approaches requires testing datasets and evaluation measurements. For BNs, datasets are usually sampled from real known networks. Otherwise, processes to randomly generate the model and the data are already established. Both practices are almost absent for RBR. Moreover, metrics to evaluate a RBN structure learning algorithm are not yet proposed.
This thesis provides two major contributions. I) A synthetic approach allowing to generate random RBNs from scratch. The proposed method allows to generate RBNs as well as synthetic relational data from a randomly generated relational schema and a random set of probabilistic dependencies. Also, we discuss the adaptation of the evaluation metrics of BNs structure learning algorithms to the relational context and we propose new relational evaluation measurements. II) A hybrid approach for RBNs structure learning. This approach presents an extension of the MMHC algorithm in the relational context. We present an experimental study to compare this new learning algorithm with the state-of-the-art approaches.

Key Words

Relational Bayesian Networks (RBN), RBN structure learning, Models generation, Evaluation metrics.