# New Data-Driven Approaches to Improve Probabilistic Model Structure Learning

**Jianjun Zhao**

**School of Computer Science and Engineering**

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

**2019**

# Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

Jan. 2019

.....

Date

Jianjun Zhao

.....

Jianjun Zhao

# Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

Nov. 2019

.....

Date

.....

Prof. Sinno Jialin Pan

# Authorship Attribution Statement

This thesis contains material from 2 papers published in the following peer-reviewed journals in which I am listed as an author.

Chapter 3 is published as Zhao, Jianjun, and Shen-Shyang Ho. Improving Bayesian network local structure learning via data-driven symmetry correction methods. International Journal of Approximate Reasoning 107 (2019): 101-121. DOI: 10.1016/j.ijar.2019.02.004

The contributions of the co-authors are as follows:

- I provided the initial research idea and prepared the manuscript drafts. The manuscript was revised by Prof Ho.
- I analyzed the results and derived the conclusions with Prof Ho.
- According to the suggestions from the reviewers, Prof Ho help me to clarify the algorithms and elaborate the content.

Chapter 5 is published as Zhao, Jianjun, and Shen-Shyang Ho. Structural knowledge transfer for learning Sum-Product Networks. Knowledge-Based Systems 122 (2017): 159-166. DOI: 10.1016/j.knosys.2017.02.005.

The contributions of the co-authors are as follows:

- Prof Ho provided the guidance on the research direction and high-level connections to other papers. He also revised and edited the manuscript drafts.
- I prepared the manuscript drafts, proposed the transfer methods and wrote the codes for the experiments.
- I analyzed the results with Prof Ho.

Nov. 2019 Jianjun Zhao

..... .....

Date Jianjun Zhao

# Acknowledgements

First of all, I would like to express my sincere gratitude towards my supervisor Prof. Sinno Jialin Pan and my previous supervisor Prof. Shen-Shyang Ho. Without their patient guidance, encouragement, and support, I would not be able to finish my Ph.D studies. Under their supervision, I have not only learned the state-of-the-art machine learning knowledge but also acquired a research sense to pinpoint the core ideas in a research paper and summarize the main issues in a research problem.

I would like to express my deep gratitude to the Agency for Science, Technology & Research (A*STAR) and Singapore International Graduate Award (SINGA) for providing scholarships to support my Ph.D studies in Singapore.

I would also like to appreciate all my fellow friends at the CIL Lab for their helpful discussions and friendly support: Yu Chen, Wenya Wang, Jianda Chen, Joey Zhou Tianyi, Lubos Krcal, Zehong Hu, Yaodong Yu and Sulin Liu.

Many thanks to all my friends for their companion. They include Zhu Sun, Haiyan Yin, Yang Yan, Long-Kai Huang, Chunyang Chen, Shangyu Chen, Hangwei Qian, Haiyun Peng, Xinghua Qu, Fengmao Lv, Qiang Zhou, Zichen Chen and Tianze Luo.

I am truly thankful for the excellent facilities provided by the Computational Intelligence Lab. Many thanks to the technical support from our administrator Kesavan Asaithambi.

Last but not least, I would like to sincerely thank my family, my mother, Zongkui Huang and my father, Hanchang Zhao. Special thanks for my grandfather, Yuanhao Huang, for his teaching and companion through my childhood.

# Abstract

To learn the network structures used in probabilistic models (e.g., *Bayesian network*), many researchers proposed structure learning algorithms to extract the network structure from data. However, structure learning is a challenging problem due to the extremely large number of possible structure candidates. One challenge relates to structure learning in Bayesian network is the conflicts among local structures obtained from the local structure learning algorithms. This is the so-called symmetry correction problem. Another challenge is the V-structure selection problem, which is related to the determination of edge orientation in Bayesian network.

In this thesis, we investigate the above two challenges in structure learning and propose novel data-driven approaches to overcome these challenges when building a Bayesian network. First, two new data-driven symmetry correction methods are developed to learn an undirected graph of Bayesian network. The proposed methods outperform the existing heuristic rule. Second, a weighted *maximum satisfiability* (MAX-SAT) problem is formulated to solve the V-structures selection problem. The weights are learned from data to quantify the strength of the V-structures. Our proposed solution outperforms existing methods.

Besides, we investigate how transfer learning can be used for structure learning with limited training examples and a source structure. In particular, we propose a transfer learning approach to learn the structure of a *Sum-Product Network* (SPN) which can be converted to a Bayesian network under certain conditions. Our novel approach allows one to construct the target SPN with limited training examples, given an existing source SPN from a similar domain.

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| BN | Bayesian Network |
| MAX-SAT | Maximum Satisfiability |
| SPN | Sum-Product Network |
| DAG | Directed Acyclic Graph |
| CPT | Conditional Probability Table |
| CPDAG | Completed Partially Directed Acyclic Graph |
| PC | Parents and Children set |
| MB | Markov Blanket |
| MMHC | Max-Min Hill-Climbing |
| SLL | Score-based Local Learning |
| GS | Grow-Shrink |
| IAMB | Incremental Association Markov Blanket |
| MMPC | Max-Min Parents and Children |
| SI-HITON-PC | Semi-Interleaved HITON-PC |
| PDAG | Partially Directed Acyclic Graph |
| CMB | Current Markov Blanket |
| CPC | Candidate Parents and Children set |
| TPC | Tentative Parents and Children set |
| MMMB | Max-Min Markov Blanket |
| GGSL | Graph Growing Structure Learning |
| SHD | Structural Hamming Distance |
| BIC | Bayesian Information Criteria |
| BDeu | Bayesian Dirichlet equivalent uniform |
| FGES | Fast Greedy Equivalence Search |
| GES | Greedy Equivalence Search |
| HC | Hill Climbing |
| FastCHC | Fast Constrained Hill Climbing |

| | |
|---|---|
| CNF | Conjunctive Normal Form |
| MI | Mutual Information |
| CMI | Conditional Mutual Information |
| CI | Conditional Independence |
| RAI | Recursive Autonomy Identification |
| BD | Bayesian Dirichlet |
| HPC | Hybrid Parents and Children |
| ASOBS | Acyclic Selection Ordering-based Search |
| CNN | Convolution Neural Network |
| EM | Expectation-Maximization |
| MAD | Median Absolute Deviation |

# List of Symbols

| | |
|---|---|
| $\mathbf{P}$ | Probability distribution |
| $P(\cdot)$ | Probability |
| $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \ldots$ | Set of variables |
| $X, Y, Z, \ldots$ | One-dimensional variables |
| $x, y, z, \ldots$ | Values of corresponding variables $X, Y, Z$ |
| $\mathbf{x}$ | Vector of values |
| $\mathbf{V}$ | Set of variables in the domain: $X_1, \ldots, X_n$ |
| $n$ | Number of variables, i.e., $\lvert\mathbf{V}\rvert$ |
| $\boldsymbol{\Theta}$ | Set of parameters in conditional probability tables for entire BN |
| $\mathbf{Pa_i}$ | Set of parents of $X_i$ |
| $\mathbf{G}$ | Directed acyclic graph (DAG) of a BN |
| $\mathcal{D}$ | Data set |
| $\mathbf{i}, \mathbf{j}$ | Nodes in SPN |
| $w_{ij}$ | Weight for sum node $\mathbf{i}$ and its child node $\mathbf{j}$ |
| $X - Y$ | Undirected edge |
| $X \rightarrow Y$ | Directed edge from $X$ to $Y$ |
| $X \rightarrow Z \leftarrow Y$ | V-structure with parents $X$ and $Y$, and child $Z$ |
| $vs(X, Z, Y)$ | V-structure $X \rightarrow Z \leftarrow Y$ |
| $\mathcal{S}$ | Sum-product network |
| $X_{\phi_i}$ | Scope of node $\mathbf{i}$ in SPN, i.e., the set of variables in its descendants |
| $\mathcal{Z}$ | Partition function for normalizing probability in SPN |
| $\mathcal{S}(x)$ | Probability of the instance $x$, computed at the root of SPN |
| $\mathcal{S}_i(x_{\phi_i})$ | Probability of $x_{\phi_i}$, computed at node $\mathbf{i}$ in SPN |

| | |
|---|---|
| $\bar{X}_i$ | Leaf node, the indicator function for $X_i = 1$ |
| $X'_i$ | Leaf node, the indicator function for $X_i = 0$ |
| $\mathcal{C}$ | Constraint-based local structure learning algorithm |
| $\mathcal{SC}$ | Symmetry correction method |
| $\mathcal{PC}$ | Parents and children set, i.e., set of direct neighbors |
| $\mathcal{MB}$ | Markov blanket |
| $\mathcal{LS}$ | Local structure, $\mathcal{PC}$ or $\mathcal{MB}$ |
| $\mathcal{SK}$ | Skeleton of BN |
| $\mathcal{G}, \mathcal{G}_V$ | Partial DAG |
| $\overline{\mathcal{LS}}$ | Symmetry corrected local structure |
| $\overline{\mathcal{PC}}$ | Symmetry corrected parents and children set |
| $\overline{\mathcal{MB}}$ | Symmetry corrected Markov blanket |
| $\mathcal{CPC}$ | Candidate parents and children set |
| $X \not\perp Y$ | X and Y are dependent |
| $X \perp Y$ | X and Y are independent |
| $X \not\perp Y \vert \mathbf{S}$ | X and Y are dependent upon conditioning on (at least one value assignment of) the variables in $\mathbf{S}$ |
| $X \perp Y \vert \mathbf{S}$ | X and Y are conditionally independent given $\mathbf{S}$ |
| $\mathcal{A}$ | Asymmetric pairs of nodes |
| $\mathcal{G}_0$ | Starting DAG |
| $\mathcal{G}_l$ | Local DAG |
| $\mathcal{E}$ | Set of asymmetric pairs confirmed to be existent |
| $\mathcal{N}$ | Set of asymmetric pairs confirmed to be non-existent |
| $\mathbf{S}_{X_i,X_j}$ | Sep-set for node $X_i$ and $X_j$ |
| $b$ | Size of conditioning set |
| $\text{Diff}_{\mathcal{SC}}$ | Symmetrical difference between the true PC(MB) and the learned PC(MB) with symmetry correction $\mathcal{SC}$ |
| $a$ | Number of asymmetric pairs |
| $c_{\mathcal{SC}}$ | Number of wrongly corrected pairs using symmetry correction $\mathcal{SC}$ |

| | |
|---|---|
| $D_{\mathcal{SC}}$ | Closeness of the learned PC(MB) with symmetry correction $\mathcal{SC}$ to the true network, compared to the AND-rule method |
| $\alpha$ | Significance level |
| $\Gamma(x)$ | Gamma function |
| $d$ | Data for the local model |
| $g_1$ | Dependent model |
| $g_2$ | Independent model |
| $\Pi_i$ | Parent set of $X_i$ |
| $c_{ijk}$ | Counts of $X_i = k$ when its parent set $\Pi_i = j$ |
| $\alpha_{ijk}$ | Hyperparameters of the Dirichlet priors |
| $q_i$ | Number of instances of $\Pi_i$ |
| $r_i$ | Number of values of $X_i$ |
| $\bar{\alpha}$ | Imaginary or equivalent sample size |
| $B_1, B_2$ | Bayes factors |
| $V_i$ | Boolean variable representing the existence of $i$-th V-structure |
| $v_i$ | Unit clause for $V_i$ = True |
| $\neg v_i$ | Unit clause for $V_i$ = False |
| $w$ | Weight for the unit clause |
| $w_i$ | Weight for $v_i$ |
| $\hat{w}_i$ | Weight for $\neg v_i$ |
| $\mathcal{W}$ | Weighting method |
| $\mathcal{VS}_{selected}$ | V-structures selected according to the solution of weighted MAX-SAT solver |
| $\mathcal{VS}_{all}$ | All possible V-structures on $\mathcal{SK}$ |
| $k$ | Number of data instances |
| $\mathcal{X}$ | Feature space of data |
| $\mathcal{D}_S$ | Source domain |
| $\mathcal{D}_T$ | Target domain |
| $\mathcal{S}_S$ | SPN learned from the source training data |
| $\mathcal{S}_T$ | SPN learned from the target training data and auxiliary knowledge from $\mathcal{S}_S$ |
| $\mathcal{X}_S$ | Feature space of the source data |
| $\mathcal{X}_T$ | Feature space of the target data |

| | |
|---|---|
| $k_S$ | Number of the source data samples |
| $k_T$ | Number of the target data samples |
| $C$ | Cluster variable |
| $c_1, c_2$ | Cluster labels |
| $\mathbf{m_i}$ | Centroid of the $i$-th cluster |
| $\mathbf{T}$ | Set of instances |
| $\mathbf{V}_j$ | Subsets of variables |
| $\mathbf{T}_i$ | Subsets of instances |
| $\mathbf{V}_{ignore}$ | Subsets of ignored variables |
| $M_i$ | Modified Z-score for $X_i$ |
| $\mathbf{p_t}$ | Vector of the marginal probability for the target data |
| $\mathbf{p_s}$ | Vector of the marginal probability for $\mathcal{S}_S$ |
| $\mathbf{d}$ | Difference between $\mathbf{p_t}$ and $\mathbf{p_s}$ |
| $m$ | Number of nodes in $\mathcal{S}_S$ |
| $K$ | Number of clusters in the top layer of $\mathcal{S}_S$ |
| $t$ | Data instance in $\mathbf{T}$ |
| $\mathbf{Q}$ | Set of query variables |
| $\mathbf{E}$ | Set of evidence variables |

# Chapter 1

# Introduction

## 1.1 Motivation

Probability plays an important role in modeling since it can properly handle uncertainty. In contrast to the deterministic models, which only give a single outcome, the probabilistic models provide a probability distribution as a solution. Probability distributions and random variables used in probabilistic models enable the modeling of uncertainty. But naive probabilistic modeling on the joint probability distribution of all variables is not computationally affordable because of the "the curse of dimensionality" problem. Hence, the probabilistic models require certain network structures, which can simplify the relations between variables and reduce the dimension in the model. The structure of a model sometimes can be constructed manually using the domain knowledge. However, domain knowledge is not always available. Hence, we have to learn the structure from the data. Moreover, if the network structure is simplified too much, the model will not be able to capture useful relations due to its limited capacity. This is not desirable as it loses the relations we would like to model. A good structure can keep important relations while ignoring the insignificant ones.

Because the space of network structures is usually huge, finding an optimal structure is a difficult and challenging task. This drives the interest in structure learning, which attempts to learn good structures efficiently and effectively from data. After studying many structure learning algorithms, we identify the key limitation of existing works: they did not fully utilize the given data. In particular, data is only

directly used in the earliest stage of the algorithms but not in the later stages. Here, a stage represents a group of steps for a sub-task in the algorithm. The following stage only takes the intermediate structure results produced by the earliest stage as input. For example, in our first study, some algorithms for Bayesian network structure learning learn the neighborhood of each node as their intermediate structure results from data [1, 2]. The heuristic symmetry correction rule is applied to the neighborhoods in the next stage. Then the symmetry corrected neighborhoods are used to construct the skeleton, i.e., the undirected graph. The process is shown in Figure 1.1. In contrast, our data-driven symmetry correction methods use data to correct the learned neighborhoods as shown in Figure 1.2.

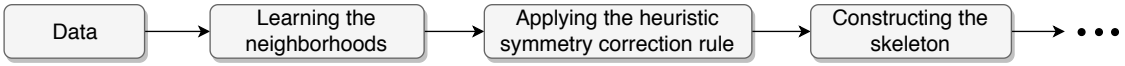



FIGURE 1.1: The workflow chart of some Bayesian network structure learning algorithms that apply the heuristic symmetry correction rule.

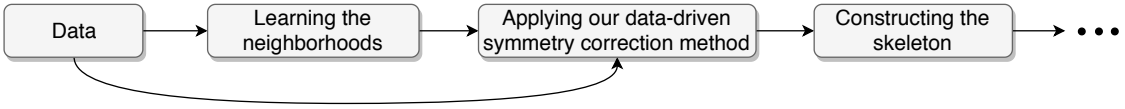



FIGURE 1.2: The workflow chart of Bayesian network structure learning algorithms with our data-driven symmetry correction method.

Due to the difficulty and complexity in structure learning, even if the intermediate structure results are extracted from data, data can still be useful in the later stages of the algorithm. For instance, one of the later stages in some algorithms for Bayesian network structure learning is learning the edge orientation. Data is directly used to orient edges in the skeleton, where the skeleton is the intermediate structure result from the previous stage. We attempted to deepen the idea of multiple uses of data in the algorithm. We believed and have shown that such multiple uses of data at different stages in the algorithm can be extended further and used to improve the overall quality of the structure learning.

The workflow chart in Figure 1.2 is similar to the classifier chain models used in multi-class and multi-label classification tasks [3]. The classifier chain models try to find the optimal order for all the classification tasks. Easy ones are done first followed by the hard ones. In our structure learning process, data is used in the earliest stage to obtain the intermediate structure(s). The feed of both the data and the intermediate structure(s) into the later stages is similar to the classifier chain models. However, the difference is that in our structure learning the order

cannot be changed and optimised like those in classifier chain models, since the later stages in our structure learning depend on the early stage and cannot be re-ordered.

Multiple uses of data are actually not new and have been applied to model learning. Many probabilistic models, including Bayesian network and sum-product network, require both structure learning and parameter learning as components in learning the model. Data is first used to find a good structure and then further used to find the parameters by fitting the learned structure. The same set of data is used twice. This shows that in order to learn a model, it is common to use data to learn different components of the model, especially learning a complex model.

## 1.2 Research Problems

Bayesian network (BN) has been widely used in many tasks for modeling and inference. It is a compact and interpretable representation of a joint probability distribution. A directed acyclic graph (DAG) represents the structure component of BN, which is the target of structure learning. In the learning process, some BN structure learning algorithms learn the neighborhood of each node to construct the skeleton. The conflict between two neighborhoods is the so-called symmetry correction problem which we will first investigate in this thesis. The V-structure selection for edge orientation is our second research problem. It appears after we have obtained the skeleton from the early stages of the learning algorithm. The previous heuristic method also reuses data but we found it only considers the conflicts among the V-structures greedily and may fail under certain conditions, where V-structure is a key structure in BN and contains two directed arcs. So we work on the edge orientation problem to improve the selection of V-structures for the skeleton.

For the above two research problems, we reuse the same training data for learning the BN structure. On the other hand, transfer learning techniques have been developed for BN to transfer the structure or data in the source domain to the target domain to help the structure learning in the target domain. This can be viewed as multiple uses of data across domains. Moreover, a probabilistic model that is closely related to BN, named Sum-Product Network (SPN), recently attracts

much interest. SPN structure learning requires a large amount of data. When only limited target data is available, the learned SPN does not perform well. Our third research study investigates the use of transfer learning to improve the SPN structure learning.

## 1.3 Research Contributions

Our data-driven methods aim to better handle the three research problems than existing heuristic methods. By making better use of data, we can achieve much better performance. Our major contributions in this dissertation are summarized as follows:

- We investigate the symmetry correction problem which exists in the Bayesian network local structure learning algorithms and propose two novel data-driven symmetry correction methods. The symmetry correction is a middle stage of the algorithms to deal with the conflict of local structures. Data is used to perform a conditional test or a search of graphs among the nodes involving each conflict. The result from the test or the search is used to resolve the conflict. Our methods handle the conflicts more reasonably, compared to existing simple heuristic rule. Our proposed methods achieve better symmetry corrections and hence, produce a better Bayesian network.

- We propose a successful formulation of weighted MAX-SAT for V-structures selection in the edge orientation of Bayesian network structure learning. We also propose two weighting methods that measure the strength of V-structures in terms of the dependence and conditional dependence on each V-structure. The previous tests of V-structures are performed separately for each V-structure. On the other hand, we test the V-structures jointly with weights and constraints. And for a particular V-structure, the data of other V-structures can help to decide the existence of that V-structure. We obtain a better selection of V-structures, compared to previous methods.

- We propose a novel approach that allows one to learn and utilize a sum-product network with limited training examples, given a sum-product network derived from a similar domain. Our proposed transfer learning approach

first utilizes the top layer clustering information on the source SPN structure to lay the foundation for the target SPN structure. Then, the deeper layers in the SPN structure are learned with the target data. We can take the learning of the top layer of SPN as one stage and learning the deeper layers as another one. With transfer learning, we do not simply reuse the target data at those two stages. Instead, we use the target data in the later stage based on the top layer structure of the source SPN, which is learned with the source data. This is because limited target data cannot learn a good top layer of the SPN. This structural transfer improves the performance of the learned SPN.

## 1.4 Dissertation Organization

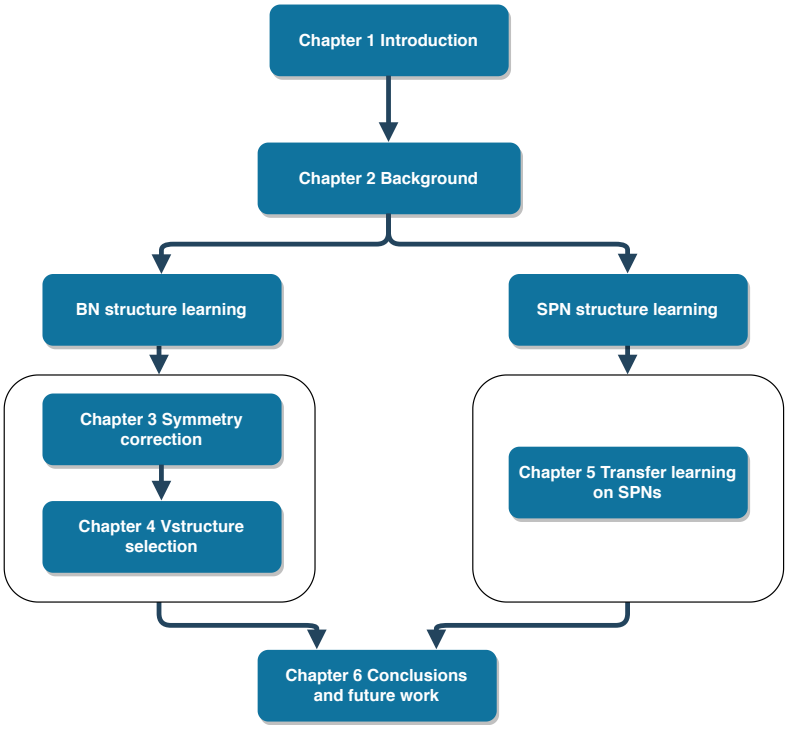



FIGURE 1.3: The content arrangement of the thesis.

The goal of this thesis is

*to design and develop new data-driven techniques for improving the existing structure learning algorithms of probabilistic models derived from data, as well as demonstrate their utility from multiple uses of data at later stages in the algorithms.*

The dissertation is organized as follows and shown in Figure 1.3:

In Chapter 2, we provide the background for BN and SPN, including an overview of BN structure learning and the inference in SPN. In Chapter 3 and 4, by analyzing the procedures of existing algorithms, we propose two improvements for BN, which use the training data multiple times, at the later stages of the structure learning to obtain better intermediate structure results. Firstly, we propose two data-driven symmetry correction methods to better handle the inconsistency in the local structures for learning the skeleton, described in Chapter 3. Secondly, we design two weighting methods and build a formulation to incorporate V-structures and their constraints as a weighted MAX-SAT problem for learning the edge orientation of a known skeleton, described in Chapter 4. In Chapter 5, we demonstrate that transfer learning techniques can be applied to improve the SPN structure learning algorithm, LearnSPN. Finally, Chapter 6 concludes this dissertation and points out three promising directions for future work.

# Chapter 2

# Background

In this chapter, we provide some background knowledge of Bayesian network and sum-product network, two representatives of probabilistic models, and an overview of the Bayesian network structure learning algorithms, since we will investigate the structure learning for Bayesian network in Chapter 3 and 4 and the transfer learning on the structure learning for sum-product network in Chapter 5. This background knowledge serves as a basis for understanding the problem that this dissertation tries to solve. Specifically, we first make a brief introduction to Bayesian network and then move to different types of Bayesian network structure learning. For the sum-product network, we introduce the basic definitions and explain how the inference is performed.

## 2.1 Bayesian Network

### 2.1.1 Bayesian Network Model

A Bayesian network (BN) represents the probability distribution $\mathbf{P}$ of a set of $n$ variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ [4]. It contains a *directed acyclic graph* (DAG) $\mathbf{G}$ which represents its structure, and a set of *conditional probability tables* (CPTs) $\mathbf{\Theta}$ which represents its parameters. The nodes in the DAG represent the variables in the distribution while the arcs between nodes represent the possible dependency and determine how the CPTs are constructed. The CPTs store the probability of each state given the parents of the target node, $P(X_i|\mathbf{Pa}_i)$, where $\mathbf{Pa}_i$ is the set

of the parents of $X_i$ in $\mathbf{G}$. From this, the joint distribution $P(\mathbf{X})$ is factorized as follows:

$$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i|\mathbf{Pa}_i) \tag{2.1}$$

In Figure 2.1, an example of Bayesian network is shown. It models the probability dependencies among four binary random variables: *Cloudy*, *Sprinkler*, *Rain* and *WetGrass*. Each row of each CPT records a set of the probabilities that sum up to one. For a particular combination of values ("True" or "False") of the parents of a variable, the probabilities of the variable taking all possible values ("True" and "False") are listed in the corresponding entries. For example, given that it is cloudy, the probability of raining is 0.8.

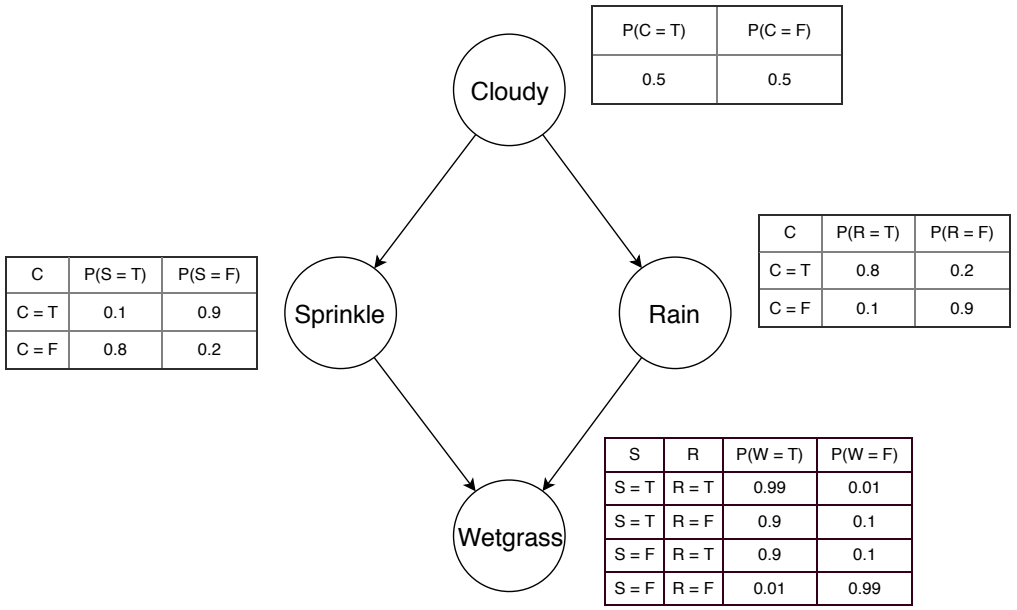



FIGURE 2.1: An example of Bayesian network with four binary variables.

Node $X$ is a *parent* of Node $T$ and then node $T$ is a *child* of node $X$ if there is an arc from node $X$ to node $T$. Nodes $X$ and $Y$ are *spouses* of each other if they have a common child and there is no arc between $X$ and $Y$. Nodes $X, Y$ and $Z$ form a *V-structure* if $X$ and $Y$ are spouses and $Z$ is their common child. The *skeleton* of a BN is obtained by stripping the orientation of the DAG. A V-structure $X \rightarrow Z \leftarrow Y$ implies one dependence and one independence based on its structural properties. One is that given the common child $Z$, $X$ and $Y$ are dependent and the other one is that $X$ and $Y$ are independent. V-structures are crucial to the structure of a BN. From Theorem 2.1.1, the skeleton and V-structures uniquely decide an equivalent class, which can be represented by a completed partially directed acyclic graph

(CPDAG), covering a set of DAGs. CPDAG may have directed and undirected edges. CPDAG has edge $X \to Y$ if and only if the edge $X \to Y$ is common to all DAGs in its equivalence class. If the class contains a DAG with $X \to Y$ and a DAG with $X \leftarrow Y$, then the CPDAG has the undirected edge $X - Y$.

**Theorem 2.1.1.** [5] Two DAGs are equivalent if and only if they have the same skeleton and the same V-structures.

If $\mathbf{X}, \mathbf{Y}$ and $\mathbf{Z}$ are three disjointed subsets of nodes in a DAG $\mathbf{G}$, then $\mathbf{Z}$ is said to *d-separate* $\mathbf{X}$ from $\mathbf{Y}$ if along every path between a node in $\mathbf{X}$ and a node in $\mathbf{Y}$ there is node $W$ satisfying one of the following two conditions: (1) $W$ has converging arrows, and none of $W$ or its descendants are in $\mathbf{Z}$, or (2) $W$ does not have converging arrows and $W$ is in $\mathbf{Z}$ [4]. The d-separation relationship of $\mathbf{X}$ and $\mathbf{Y}$ given $\mathbf{Z}$ can be read from the DAG $\mathbf{G}$. On the other hand, within a distribution $\mathbf{P}$, there hold conditional independence relations among variables or sets of variables. A DAG $\mathbf{G}$ and a distribution $\mathbf{P}$ are *faithful* to one another when the d-separation statements of $\mathbf{G}$ and the independence statements of $\mathbf{P}$ are equivalent [6].

For each node in the DAG, there are local structures that describe the relationship between the target node and its neighboring nodes. The local structures consist of the *parents and children* set (PC) and the *Markov Blanket* (MB) [4]. The nodes in the PC are also known as *neighbors* because the parent nodes and child nodes are the only nodes that have direct arc connections to the target node. The MB of a target variable $T$ is defined to be a set of nodes conditioned on which all other nodes are independent of $T$ [4]. In BNs, the MB consists of PC and spouses, since by conditioning on it, all other nodes are independent of the target node, due to the structure of DAG.

### 2.1.2 Overview of Structure Learning

Compared to the well-studied parameter learning of BN given a known DAG [7], the structure learning is a challenging task. Since the exact BN structure learning is an NP-hard problem [8], there is no algorithm that can scale to a large number of variables. However, it is worth noting that there are still exact search algorithms proposed to find the optimal DAG for a certain score criterion. A dynamic programming approach decomposes the task of finding the best DAG for a variable

set $\mathbf{W} \subseteq \mathbf{V}$ into three sub-tasks, which are finding the best sink $S \in \mathbf{W}$, finding the best parents for the sink $S$, and finding the best DAG for $\mathbf{W} \setminus S$ [9]. By using this decomposition repetitively, the algorithm finds an optimal ordering that is compatible with the best DAG. Given such an optimal ordering, it is much easier to find the best DAG. The ordering limits the candidate parents for each node since only the predecessors of a node in the ordering can be its parents. Moreover, integer programming is also used to solve the search over graph structures [10, 11]. The structure learning problem is formulated as a linear program over a polytope, which represents DAGs. Hence, various integer programming solving techniques can be applied and then the solution is interpreted back to BN to get an optimal DAG.

In addition, the tree-structured BN, a structure where each node has at most one parent, can be solved optimally by the Chow-Liu algorithm [12]. The Chow-Liu algorithm constructs the approximate distribution that has the minimum Kullback-Leibler divergence to the actual distribution, by finding the maximal spanning tree, with weights from the mutual information.

The algorithms above find the optimal DAG but they also have their limitations. The exact search methods cannot handle a large number of variables and the Chow-Liu algorithm restricts the learned BN to be tree-structured. To address the problem of learning the general BN with a large number of variables, heuristic local algorithms [2] are proposed, as they can scale to high-dimensional data sets without structure restrictions. Another approach is the heuristic search algorithms, which can scale to high-dimensional data sets as well. There are three types of algorithms for BN structure learning, namely: constraint-based algorithms, score-based algorithms, and hybrid algorithms.

Constraint-based algorithms learn the network structure by analyzing the probabilistic relations using conditional independence tests on the training data [13]. It then uses these relations as *constraints* to find the network structures that are consistent with these constraints by removing or keeping edges in the network. There are many constraint-based algorithms proposed, such as the SGS [14], the PC algorithm [15], and Max-Min Parents and Children (MMPC) algorithm [1].

On the other hand, score-based algorithms perform some heuristic search, such as hill-climbing and tabu search [16] in the space of graph structures. It assigns a

score to each candidate DAG $\mathbf{G}$ and this score describes how well the data $\mathcal{D}$ fits $\mathbf{G}$. The score is the posterior probability of $\mathbf{G}$ given $\mathcal{D}$. By the Bayes' rule,

$$\text{Score}(\mathbf{G}, \mathcal{D}) = \text{P}(\mathbf{G}|\mathcal{D}) = \frac{\text{P}(\mathcal{D}|\mathbf{G})\text{P}(\mathbf{G})}{\text{P}(\mathcal{D})} \tag{2.2}$$

$\text{P}(\mathcal{D}|\mathbf{G})$ is the likelihood of $\mathcal{D}$ given $\mathbf{G}$ and $\text{P}(\mathbf{G})$ is the prior over the DAGs. During the search, the score is used to evaluate the candidate DAGs, and it is maximised to find the optimal DAG. We only need to maximise the numerator, since the denominator does not depend on $\mathbf{G}$. Widely-used scores include the *Bayesian Information Criterion* (BIC) [17] and *Bayesian Dirichlet equivalent uniform* (BDeu) [18]. As the search is a greedy search, it may be trapped at some local optimal point and cannot reach the global optimal point. A common space of graph structure would be the space of DAGs. The allowed operators on DAG are arc addition, arc deletion, and arc reversal. Another space can be used is the space of CPDAGs, which represents the equivalence class of DAGs. The greedy equivalence search (GES) performs a greedy search on the space of CPDAGs [19]. The search space of CPDAGs is smaller than the space of DAGs and the search steps between equivalent DAGs are saved because of the equivalence class.

Furthermore, a hybrid algorithm is a combination of score-based and constraint-based algorithms. One popular type of hybrid algorithms learns the skeleton by using the constraint-based algorithm and performs search on the space restricted by the skeleton. An example of such hybrid algorithms is the max-min hill-climbing algorithm (MMHC) [20].

## 2.2 Sum Product Network

### 2.2.1 Sum Product Network Model

A Sum-Product Network (SPN) is a probabilistic model with a deep architecture [21]. An SPN $\mathcal{S}$ is represented by a rooted, directed acyclic graph (DAG) with internal sum nodes and product nodes over a set of random variables $\mathbf{V} = \{X_1, \ldots, X_n\}$. The leaves of SPNs are univariate distributions and there is a positive weight $w_{ij}$ attached to each edge between sum node $\mathbf{i}$ and its corresponding

child node **j**. $\mathcal{S}(x)$ is the unnormalized probability of the instance $x$, i.e., the value of the root when the network is evaluated bottom-up, given $X = x$.

The scope of a node **i**, $X_{\phi_i}$ is the set of variables in its descendants. An SPN is decomposable if the scopes of the children of product nodes are all disjointed. An SPN is complete if the scopes of the sum node's children are the same. If an SPN is decomposable and complete, then it is *valid* [21]. Validity, together with normalization conditions, $\sum_j w_{ij} = 1$ and the normalized leaf distribution ensures that evidence probability can be calculated by evaluating the network, i.e. $P(X = x) = \mathcal{S}(x)$ and the partition function $\mathcal{Z}$, define as $\mathcal{Z} = \sum_x \mathcal{S}(x)$ becomes 1. In the evaluation of the probability of an instance, for a leaf node **i** representing variable $X_k$, $\mathcal{S}_i(x_k) = P(X_k = x_k)$. For a sum node **i**, $\mathcal{S}_i(x_{\phi_i}) = \sum_j w_{ij}\mathcal{S}_j(x_{\phi_j})$ and for a product node **i**, $\mathcal{S}_i(x_{\phi_i}) = \prod_j \mathcal{S}_j(x_{\phi_j})$, where node **j** is a child of node **i** and $\phi_i, \phi_j$ are corresponding sets of variables. For example, a valid SPN containing four components over binary variables $X_1$ and $X_2$ is shown in Figure 2.2. The root is a sum node and its weights sum up to 1. The product nodes combine variables $X_1$ and $X_2$. The leaf nodes $\bar{X}_1$ and $X'_1$ represent Bernoulli distributions for variable $X_1$ with different parameters while nodes $\bar{X}_2$ and $X'_2$ are for variable $X_2$.

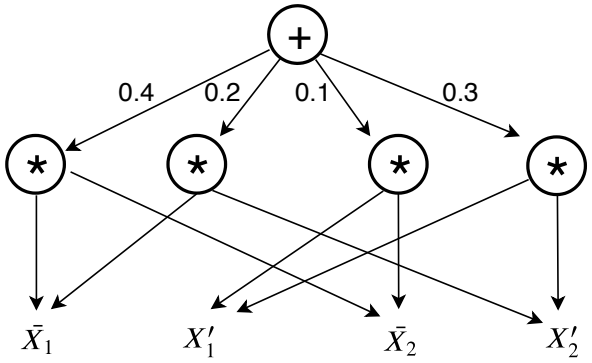



FIGURE 2.2: An example of a valid SPN containing four components over binary variables $X_1$ and $X_2$.

### 2.2.2 Inference

One major advantage of an SPN is its capability of performing exact probabilistic inference, which enables the accurate and efficient query of conditional probability

and marginal probability. For example, we can easily compute the probability $P(X_1 = 1, X_2 = 1)$ in Figure 2.2 by setting the bottom leaf nodes ($\bar{X}_1 = 1, X'_1 = 0, \bar{X}_2 = 1, X'_2 = 0$). The product nodes in the middle layer compute the product of the corresponding leaf nodes. Only the leftmost product node gives 1 while others give 0. And the top sum node returns the probability, $0.4*1+0.2*0+0.3*0+0.1*0 = 0.4$. We can also compute the marginal probability $P(X_1 = 1)$ by setting the leaf nodes ($\bar{X}_1 = 1, X'_1 = 0, \bar{X}_2 = 1, X'_2 = 1$). Similarly, the values are passed upwards and the probability is $0.4 * 1 + 0.2 * 1 + 0.3 * 0 + 0.1 * 0 = 0.6$. We can compute the conditional probability by using the joint probability and marginal probability.

$$P(X_1 = x_1 | X_2 = x_2) = \frac{P(X_1 = x_1, X_2 = x_2)}{P(X_2 = x_2)} \tag{2.3}$$

Given an SPN, the exact inference can be performed efficiently by feeding the data instance to the leaf nodes. In fact, inference and modeling are performed together. This overcomes performance degradation due to approximate inference used in intractable models, such as Bayesian networks [21]. Furthermore, an SPN has a deep architecture which allows for more expressiveness [22, 23] and much more efficient representation compared to shallow architecture [24]. On the other hand, the SPN structure can be large and complex, which requires a large amount of data and computation time to achieve an accurate structure.

# Chapter 3

# Improving Bayesian Network Local Structure Learning via Data-driven Symmetry Correction Methods

As mentioned in Chapter 2, there are constraint-based algorithms and score-based algorithms proposed for Bayesian network structure learning. These algorithms learn the local structures for each node separately, which are more efficient than learning the structure for all nodes globally. Constraint-based local algorithms [1, 2, 25, 26] and some recently proposed score-based local algorithms [27, 28] have shown their capability of using learned local structures to construct the skeleton. However, most of these algorithms suffer from the *symmetry correction problem* [2, 20] when they build the Bayesian network from the local structures. This problem arises when there are conflicts in the separately learned local structures.

This chapter proposes two data-driven methods to solve the symmetry correction problem: a score-based method **symG** and a constraint-based method **symC**. To decide the existence of the link between a pair of nodes, our score-based method **symG** utilizes the Greedy Hill-Climbing search over the union of the nodes of the local structures of the pair of nodes. On the other hand, our constraint-based method **symC** performs conditional tests over those nodes. Our experimental results show that for constraint-based algorithms, the symmetry correction problem

can be better handled by **symG**, compared to AND-rule and **symC**. Moreover, the performance of a score-based local learning algorithm (e.g., SLL [27]) is better when **symC** is used, compared to using **symG**. We observe that the combination of a learning algorithm of a particular type (i.e., constraint-based or score-based) and a symmetry correction method of another type improves the overall performance.

This chapter is from our published paper [29] and organized as follows: we first introduce the local structure algorithms that require the symmetry correction in Section 3.1; we then describe the symmetry correction problem and the AND-rule, a commonly used heuristic rule to solve the symmetry correction problem in Section 3.2; we propose our symmetry correction methods and describe how they work in Section 3.3, followed by the empirical evaluation in different settings in Section 3.4. Finally, in Section 3.5 we conclude this chapter.

## 3.1 Local Structure Learning

A constraint-based algorithm or a score-based algorithm can be categorized as either a global or a local algorithm. A global algorithm takes into consideration all the variables in the dataset when building the BN. For example, a score-based global algorithm performs a search to find the whole network. On the other hand, a local algorithm learns the local structures for each node separately, which usually only involves a limited number of variables and then uses them to build the whole network structure. The advantage of the local structure method is efficiency. The global method considers the relations among all nodes. The computational cost rises exponentially as the size of the nodes increases. This limits the network size when using global methods. In contrast, the local methods focus on the target node. If another node depends on the target node given its current local structure, then it can be added into the local structure. Otherwise, it would be ignored. So for each target node, only a small number of nodes often are considered, which greatly improves efficiency.

Since only local algorithms contain the symmetry correction problem, we focus on local algorithms in this chapter. In particular, we focus on constraint-based ones. This is because there are many proposed constraint-based local algorithms.

Recently, a few score-based local algorithms have been proposed. We will look at the symmetry correction problem for one of them.

### 3.1.1 Constraint-based Local Structure Learning

For the constraint-based algorithms, there are global algorithms such as the SGS [14] and the PC algorithm [15], which learn the whole Bayesian structure. There are also local algorithms such as Grow-Shrink (GS) algorithm [25], Incremental Association Markov Blanket (IAMB) algorithm [26], Max-Min Parents and Children (MMPC) algorithm [1] and semi-interleaved HITON-PC (SI-HITON-PC) [2]. These constraint-based algorithms can be summarized in three stages, which are learning Markov blankets (optional), learning neighbors and learning arc directions [30]. They learn the local structures, MB and PC, to construct the BN structure.

In Algorithm 1, we show how the general constraint-based BN structure learning algorithm works. After we input data and choose a learning algorithm and a symmetry correction algorithm, we need to check that the local structure learned from the algorithm is MB or PC (line 1). If it is PC, then we apply the symmetry correction and construct the skeleton (lines 2,9). If it is MB, then we apply the symmetry correction to get corrected MBs (line 2). By using these corrected MBs, we find the PCs and we apply the symmetry correction to some of these PCs, which contain asymmetric pairs (lines 3-8). The number of the asymmetric pairs is usually limited due to the corrected MB. The corrected PCs are used to construct the skeleton (line 9). After obtaining the skeleton, we learn and apply the V-structures (line 10). The last step is to orient other possible edges while avoiding new V-structures and directed cycles (line 11). Finally, the algorithm returns the learned partial DAG (PDAG).

In contrast to global algorithms, such as the PC algorithm [15], which can only handle a limited number of variables in BN, the GS algorithm [25] as a local learning algorithm can learn much larger networks. It follows steps in Algorithm 1. It learns the graph structure by first learning the MB and then using the MB to find the PC. Finally, based on the MB and PC, a PDAG is constructed after the determination of edge orientations and acyclicity checks. The algorithm usually requires the existence of a faithful DAG and it uses the following theorem to determine the PC.

**Algorithm 1: General constraint-based BN structure learning algorithm**

**Input:** constraint-based local structure learning algorithm $\mathcal{C}$, dataset $\mathcal{D}$, symmetry correction method $\mathcal{SC}$.
**Output:** Partial DAG $\mathcal{G}$.

1: Learn the local structure $\mathcal{LS}$, which can be the Markov blanket $\mathcal{MB}$ or the parent-child set $\mathcal{PC}$, of each node $X_i$ from $\mathbf{V}$, the set of nodes, using a constraint-based local structure learning algorithm $\mathcal{C}$ and dataset $\mathcal{D}$.
2: Perform symmetry correction $\mathcal{SC}$ on $\mathcal{LS}$ to get symmetry corrected local structure $\overline{\mathcal{LS}}$ for each node $X_i$, if necessary.
3: **if** $\mathcal{LS}$ is the Markov blanket **then**
4:  **for** $X_i \in \mathbf{V}$ **do**
5:  Use $\overline{\mathcal{LS}}(X_i)$ to find the $\mathcal{PC}(X_i)$ and perform symmetry correction $\mathcal{SC}$ on $\mathcal{PC}(X_i)$ to get $\overline{\mathcal{PC}}(X_i)$.
6:  Set $\overline{\mathcal{LS}}(X_i)$ to be $\overline{\mathcal{PC}}(X_i)$.
7:  **end for**
8: **end if**
9: Construct the skeleton $\mathcal{SK}$ based on $\overline{\mathcal{LS}}$ from all nodes.
10: Learn the V-structures on $\mathcal{SK}$ from $\mathcal{D}$ and apply them on $\mathcal{SK}$ to get $\mathcal{G}_V$.
11: Orient other possible edges in $\mathcal{G}_V$ while avoiding new V-structures and directed cycles to get a partial DAG $\mathcal{G}$.
12: **return** $\mathcal{G}$.

**Theorem 3.1.1.** [6] Let $\mathbf{X} = (X_1, \ldots, X_n)$ be a random vector, with probability distribution $\mathbf{P}$. A faithful DAG $\mathbf{G}$ contains an edge between two distinct variables $X_i$ and $X_j$ if and only if $X_i \not\perp X_j | \mathbf{S}$ for any $\mathbf{S} \subseteq \mathbf{V} \setminus \{X_i, X_j\}$, where $\mathbf{V} = \{X_1, \ldots, X_n\}$ is the variable set.

An edge should be removed when any of the tests conditioning on all subsets of the set $\mathbf{S}$ are found to show independence. The set $\mathbf{S}_{X_i,X_j}$ such that $X_i \perp X_j | \mathbf{S}_{X_i,X_j}$ and causes the removal of the edge $(X_i, X_j)$ is called the *sep-set*.

The first part of the GS algorithm is the GS Markov blanket algorithm, which learns the MB. It contains two phases: a growing phase and a shrinking phase. The growing phase adds any variables that are dependent on the target variable one by one into the set $\mathbf{S}$. In each iteration, the conditioning set is the previously obtained $\mathbf{S}$ and $\mathbf{S}$ starts from the empty set. The set $\mathbf{S}$ at the end of the growing phase should contain the MB. However, it may contain some variables which are not in the MB. So the shrinking phase is to remove these variables by checking if each one is independent of the target variable given the rest nodes in $\mathbf{S}$. Based on the definition of MB, the target variable is independent of the rest of the variables

given the MB. Hence, conditional independence tests are used to obtain MB from **S**. After getting the MB, the PC is constructed by testing whether one can find a sep-set such that by conditioning on it, the target node and the tested node in the PC are independent. With the help of MB, candidates of sep-set can be limited to the MB of both nodes [30]. Then a V-structure detection approach [31] is applied to the skeleton obtained from the PC, to orient the edges. Finally, the Meek's rule [32] and acyclicity constraints are applied to orient the rest of the edges to form the DAG.

The IAMB algorithm [26] also has two phases similar to the GS algorithm. The difference lies in the growing phase (phase I). The order of variables added to **S** in the GS algorithm is predefined or based on the strength of association conditioned on the empty set. However, for the IAMB algorithm, the next variable is found based on the association conditioning on the current Markov Blanket (CMB), an estimate of the MB. Hence, strongly associated variables can be added earlier to reduce the number of variables needed to remove in the shrinking phase (phase II).

The MMPC algorithm [1] has two phases as well but it focuses on learning the PC directly instead of learning the MB first. In phase I, a max-min method is used to choose variables. For each variable, a conditioning set which minimizes the association to the target variable is found. Then the variable that has the maximum association is added into the candidate parents and children (CPC) set. In phase II, if there is any independence between the target node and one node of the CPC, given all subsets of the rest of the nodes in CPC, then this node should be removed from the CPC.

MMPC only finds the PC of each node. To find the MB, the MMMB algorithm is needed. The MMMB algorithm contains the MMPC algorithm as a subroutine. Since spouses have common children with the target variable, the *fake MB*, which is a superset of the MB, can be found by applying the MMPC algorithm twice, which is the union of the PCs of the PC nodes of the target variable. In phase II, spouses are identified by testing the property that spouses are dependent on the target variable conditioning on their common children.

The semi-interleaved HITON-PC algorithm is similar to the MMPC algorithm. The inclusion step is done by prioritizing variables, throwing away independent ones, and moving the highest-priority variable to tentative parents and children

(TPC) set, a superset of PC. Every time a new variable is added to TPC, the elimination step tries to remove some weakly relevant variables. The word "interleaved" means that the inclusion and elimination steps are alternately repeated. The word "semi" means that a full variable elimination is only performed when the priority queue is empty; during other iterations only the most recently included variable is checked for elimination. The MB can be found through the same algorithm used in the Max-Min Markov Blanket (MMMB) algorithm.

In summary, all these constraint-based local algorithms contain two phases. The first one is expanding the set of candidates of MB or PC. The second phase is deleting the unnecessary variables and keeps the necessary ones in the MB or PC. Through these two phases, it learns the MB or PC. The MB and PC can be used to find each other. So the local BN structure is obtained, consisting of the MB and PC, of each variable. These local structures are then used to get the skeleton. Finally, after directing the orientation of the edges, the partial DAG is constructed.

### 3.1.2 Score-based Local Structure Learning

The score-based local learning algorithm (SLL) [27] learns the PC and MB for each node by locally applying the BN exact learning algorithm and then constructing the DAG, based on these learned MB and PC. During each iteration of the local search of each node, SLL will add one node from the rest of the unused nodes and perform the exact algorithm, dynamic programming algorithm [9], over the union of the target node, the added node and the current PC(MB) to get the optimal local DAG. The PC(MB) of the target node in the optimal local DAG is then used for the next iteration. The final PC(MB) is obtained after all nodes have been processed.

The graph growing structure learning (GGSL) [28], applying SLL as a subroutine, is a score-based algorithm as well. The difference is that it does not use the learned PC and MB of each node together but instead updates the learned local DAG into the global DAG gradually. It is an expanding process as it starts from a node to update its local DAG and then moves to the nodes in the PC of this node to update their local DAG while avoiding conflicts with previous updates. It continues until all nodes are updated. This algorithm, in a way, bridges the gap between local and global algorithms.

## 3.2 Symmetry Correction Problem

### 3.2.1 Problem Description

From the definition of PC and MB, they should be symmetric; i.e., if node $X$ is in the PC(MB) of node $T$, then $T$ should be in the PC(MB) of $X$ as well. For global algorithms, symmetry correction is not necessary, as the PC and MB are always symmetric. However, the local algorithms may produce asymmetric pairs of PC(MB) because local structure learning is applied to nodes independent of one another. For both constraint-based and score-based local algorithms, symmetric correction is required.

For example, in Figure 3.1, the PC of node $C$ and $F$ are learned from a local algorithm. $\mathcal{PC}(C) = \{A, D, F\}$ and $\mathcal{PC}(F) = \{E, H\}$. There is an asymmetric pair $(C, F)$. The problem is whether the edge between node $C$ and $F$ should be kept or not.

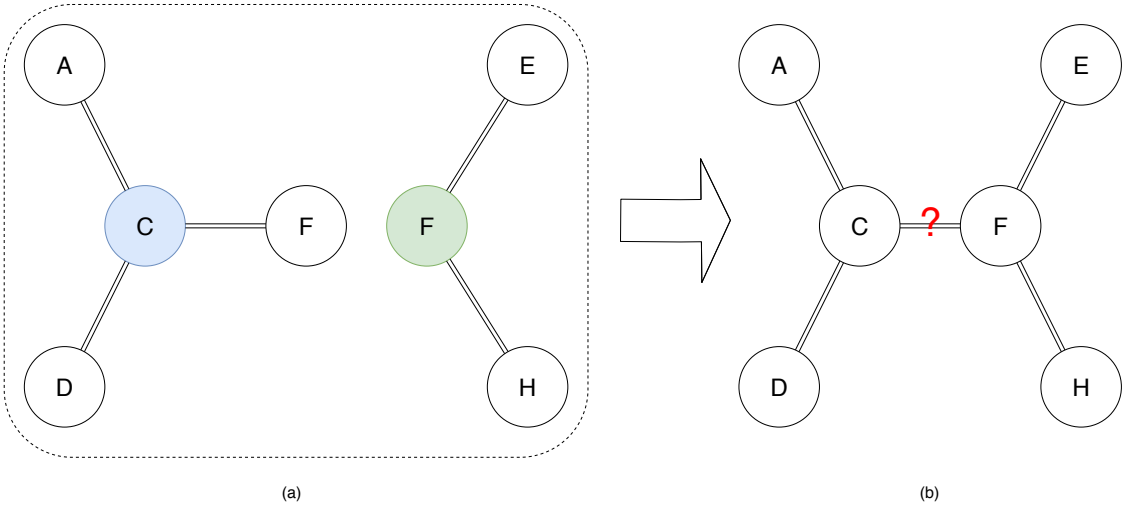



FIGURE 3.1: (a) The local structures of Node $C$ and $F$. (b) The symmetry correction problem for the integrated structure.

### 3.2.2 Previous Approach

One common method to handle the symmetry correction problem is to simply correct the asymmetric pairs by treating them as false positive and removing both nodes from the PC or MB of each other [13]. This is called the AND-rule [28].

**Definition 3.2.1. AND-rule** [28] For a node $X$ to be adjacent to $T$ in $\mathbf{G}$, the following two statements hold true: (1) $X$ must be in the PC of $T$ and (2) $T$ must be in the PC of $X$, i.e., $X \in \mathcal{PC}^{\mathbf{G}}(T)$ and $T \in \mathcal{PC}^{\mathbf{G}}(X)$.

The AND-rule is widely used in constraint-based and score-based local algorithms. For the example in Figure 3.1, after applying the AND-rule to the asymmetric pair $(C, F)$, $\mathcal{PC}(C) = \{A, D\}$, and $\mathcal{PC}(F) = \{E, H\}$. The AND-rule is simple and it fits **Theorem** 3.1.1.

Let us apply MMPC to the example in Figure 3.1. In phase II of MMPC, the CPC is already obtained, and some nodes should be removed to get the PC. The criterion for removal is to check whether the node is conditionally independent of the target node, given any subset of the rest of the CPC. Assume $\mathcal{CPC}(C) = \{A, D, F\}$ and $\mathcal{CPC}(F) = \{C, E, H\}$. For node $C$, by conditioning on any subsets of $\mathcal{CPC}(C) \setminus \{F\} = \{A, D\}$, node $C$ and node $F$ are dependent. So node $F$ is kept in the $\mathcal{CPC}(C)$. However, for node $F$, by conditioning on a subset of $\mathcal{CPC}(F) \setminus \{C\} = \{E, H\}$, (say) set $\{E\}$, node $F$ and node $C$ are independent. So node $C$ is removed from the $\mathcal{CPC}(F)$. This evidence indicates that node $E$ is a sep-set, which separates node $C$ and $F$, but it is irrelevant of what happens to $\mathcal{CPC}(C)$. So an asymmetric pair $(C, F)$ is produced from the algorithm due to separate PC learning for the two nodes. The AND-rule returning $\mathcal{CPC}(C) = \{A, D\}$ and $\mathcal{CPC}(F) = \{E, H\}$ based on this evidence may not be accurate and may lead to inaccurate DAG learned from the PC and MB.

An asymmetric pair in PC or MB occurs when there is a conflict of learned PC or MB between a pair of nodes. This indicates a possible dependency or conditional dependence between the pair of nodes. Hence, a more careful examination is needed to decide whether such dependency exists. There can be two cases: Type I error and Type II error. In the first case, the algorithm wrongly detects some dependency though the dependency does not exist. In the second case, the dependency exists, but the algorithm fails to detect it fully. As both cases are possible, we need to utilize available data to verify the existence of the dependency.

As the PC and MB relations are symmetric, the symmetry correction can be viewed as a symmetry restriction, which appears in other BN learning algorithms as well. The Constrained Hill Climbing (CHC) [33] algorithm uses a progressive restriction of the neighborhood to help the search. The restriction on arcs can be extended

to restriction on edges due to the symmetry of conditional independence [34] and this can improve the search. This restriction over edges is similar to the symmetry correction for PC and MB. Enforcing such restrictions can make the learned network more consistent and help the BN structure learning.

GGSL can avoid the symmetry correction problem as it includes the score-based local learning (SLL) to learn the local DAG and update the global DAG [28]. Since for each updating, the locally learned DAG is a proper DAG, which has symmetric PC and MB, it avoids the symmetry correction problem implicitly. However, there may be conflicts between the current global DAG and updating local DAG. Hence, some rules are used to adapt the updating local DAG to the global DAG. This can be viewed as a shift from the symmetry correction problem to the global and local DAG consistency problem. Besides, this cannot be applied to constraint-based local algorithms, because there is no local search updating to guarantee symmetries in constraint-based algorithms. Therefore, symmetry correction is still required for constraint-based local algorithms.

## 3.3 Proposed Symmetry Correction Methods

In Section 3.2, we describe how the symmetry correction problem occurs when conflicts and inconsistency arise in learned local structures. The most common approach for symmetry correction is the use of AND-rule [28], which allows the relation between a pair of nodes to exist when local structures agree on the relation. Its advantage is simplicity and consistency with the evidence from the local structures. However, the AND-rule is a heuristic approach, and it cannot accommodate uncertain conflicts in the learned local structures. These weaknesses can be overcome when the training data is used for symmetry correction. Here, we describe and discuss two novel data-driven methods to handle the symmetry correction problem: a score-based method **symG** and a constraint-based method **symC**.

### 3.3.1 Intuition

The occurrence of conflicts in local structures is similar to inconsistencies from the edge restrictions [35]. For example, a simple inconsistency of edge restriction happens when the existence and absence of the same edge occur at the same time. Similarly, in the symmetry correction problem, the existence and absence of the PC(MB) relation contradict to each other in two local structures learned using a BN local structure learning algorithm. One of the reasons for the occurrence of conflicts is that in the learning algorithms, the ordering of nodes processed for the target node affects the learned local structure of the target node. Take node $C$ in Figure 3.1 as an example. The neighbors of node $C$, $\{A, D, F\}$, can be different if we test the dependency with node $C$ in a different ordering. Suppose the ordering $\{A, D, F, E, H\}$ lead to the neighbors $\{A, D, F\}$. The ordering $\{D, A, F, E, H\}$, which tests node $D$ first, may lead to the neighbors $\{D, F\}$, because conditioning on node $D$, node $C$ and node $A$ can be tested to be independent while previously conditioning on node $A$, node $C$ and node $D$ are dependent. The change of conditioning sets along the way of testing also affects the local structure of the target node.

For existing local structure learning algorithms, training data are used only to learn the local structures with only the AND-rule heuristic correction mechanism for both constraint-based and score-based algorithms. Either of the algorithms cannot identify and eliminate conflicts when the local structures are learned.

Our proposed hybrid framework overcomes the limitation of existing BN local structure learning algorithms by integrating an alternative data-driven symmetric correction mechanism with an existing learning algorithm. The symmetric correction mechanism provides another "viewpoint" based on the same training set to correct the learned structures. For our proposed hybrid framework, a constraint-based learning algorithm is paired up with a score-based correction method. Similarly, a score-based learning algorithm is paired up with a constraint-based correction method. Towards this end, the hybrid framework can be understood as the inclusion of two different types of information to build the BN structure.

### 3.3.2 Score-based Symmetry Correction Method

For a constraint-based structure learning algorithm, an asymmetric pair of relations indicates a possible failure of the hypothesis test. This may be due to the large conditioning set. As a result, a different type of test cannot work either. Hence, a shift to the score-based method can be a better choice. Furthermore, the success of MMHC, which restricts the search space and performs a search on the skeleton, suggested that a search over local constraints is a good approach.

Constraint-based algorithms use the conditional independence test to decide whether the conditional independence $X \perp Y|\mathbf{S}$ holds or not, where $X$ and $Y$ are variables and $\mathbf{S}$ is a subset of variables. The results of the tests are called *constraints*. The graph learned by the method is the one that satisfies the most constraints. As mentioned in [6], constraint-based algorithms require the assumption of faithfulness, that the data to which the algorithm is applied is generated from distributions which have a corresponding faithful DAG. But in some cases, the DAGs are not faithful. And even if a faithful DAG exists, the power of the hypothesis test decreases rapidly as the size of the conditioning set increases. When the null hypothesis $H_0 : X \perp Y|\mathbf{S}$ is accepted, the edge $X - Y$ is removed. But for hypothesis testing, the null hypothesis is never accepted. The alternative hypothesis $H_1 : X \not\perp Y|\mathbf{S}$ is accepted when there exists an evidence to reject $H_0$. No evidence to reject $H_0$ does not mean $H_0$ is true and should be accepted.

The score-based method **symG** shown in Algorithm 2 performs a local (tabu) search [36] for a PC asymmetric pair $(X, T)$ over the union of the $PC(X)$ and $PC(T)$ in the space of DAG. The allowed operators are arc addition, arc deletion, and arc reversal. We start the search with a network $\mathcal{G}_0$ which has no arc between $X$ and $T$, and $X$ connects to the PC of $X$ with directed arcs from $X$ and so does $T$. The number of iterations the tabu search can perform without improving the best network score is set as the number of arcs in the graph. If locally learned network $\mathcal{G}_l$ finally contains an arc between $X$ and $T$, we put $X$ and $T$ into the PC of each other. Otherwise, we exclude both of them. It is worth noting that the search result is only used to decide the link of the asymmetric pair $(X, T)$. This is done separately for each asymmetric pair so the search result for pair $(X, T)$ does not affect other asymmetric pairs. Any other information obtained in the search is discarded. For example, in Figure 3.1, for the asymmetric PC pair $(C, F)$, **symG** perform a search over the nodes of their neighborhood, $\{A, C, D, E, F, H\}$. Only

**Algorithm 2: SymG**($\mathcal{LS}$, $\mathcal{D}$)

**Input:** local structures $\mathcal{LS}$ of each node obtained from a learning algorithm, where $\mathcal{LS}$ can be the Markov blanket $\mathcal{MB}$ or the parent-child set $\mathcal{PC}$, dataset $\mathcal{D}$.

**Output:** symmetry corrected local structures $\overline{\mathcal{LS}}$ of each node.

1: Extract asymmetric pairs of nodes $\mathcal{A}$ by going through all local structures $\mathcal{LS}$ of each node and checking if the pair of nodes is contained in the local structure of each other.
2: The existent set $\mathcal{E} \leftarrow \emptyset$ and non-existent set $\mathcal{N} \leftarrow \emptyset$.
3: **for** each pair of nodes $(X_i, X_j)$ in $\mathcal{A}$ **do**
4: Perform a greedy search using $\mathcal{D}$ over the union of $\mathcal{LS}(X_i)$ and $\mathcal{LS}(X_j)$ with a starting DAG $\mathcal{G}_0$, where $\mathcal{G}_0$ contains arcs from $X_i$ to $\mathcal{LS}(X_i)$ and $X_j$ to $\mathcal{LS}(X_j)$ and no arc between $X_i$ and $X_j$, and obtain a DAG $\mathcal{G}_l$.
5: **if** in $\mathcal{G}_l$, $\mathcal{LS}(X_i)$ contains $X_j$ and $\mathcal{LS}(X_j)$ contains $X_i$. **then**
6: The pair $(X_i, X_j)$ is confirmed to be existent and add it into $\mathcal{E}$.
7: **else**
8: The pair $(X_i, X_j)$ is confirmed to be non-existent and add it into $\mathcal{N}$.
9: **end if**
10: **end for**
11: Update the $\mathcal{LS}$ of each node to symmetry local structures $\overline{\mathcal{LS}}$ according to $\mathcal{E}$ and $\mathcal{N}$.
12: **return** symmetry corrected local structures $\overline{\mathcal{LS}}$ of each node.

the link between $C$ and $F$ is considered. Other links, such as the link between $A$ and $E$, are ignored. Similarly, this can be done to the MB asymmetric pair.

Our search-based method examines the potential edge directly. In GGSL, the determination of the edge is more implicit. The number of such searches depends on the number of asymmetric pairs, but it is usually only a portion of the total number of pairs of nodes in the DAG as shown in Table 3.2. This also depends on the data size and the algorithm used, but this means that we do not need to perform the search for every pair of nodes during symmetry correction. The computation cost in enforcing the AND-rule is high in SLL, but it can be reduced [37]. Here, we care more about the cost of a single symmetry correction and the number of symmetry corrections required instead of the cost for the exact BN learning due to symmetry restriction.

### 3.3.3 Constraint-based Symmetry Correction Method

Score-based structure learning algorithms evaluate a few candidate structures and choose the best one based on a score criterion. An asymmetric pair resulted from two local searches indicates that one or both of them get stuck at some local optimal structure, especially when the search space is large compared to the number of visited structures. An additional search over the local structures may not be very helpful, but a constraint-based method can be an alternative to get around with those sub-optimal points, as it only considers the conditional dependence between the pair of nodes.

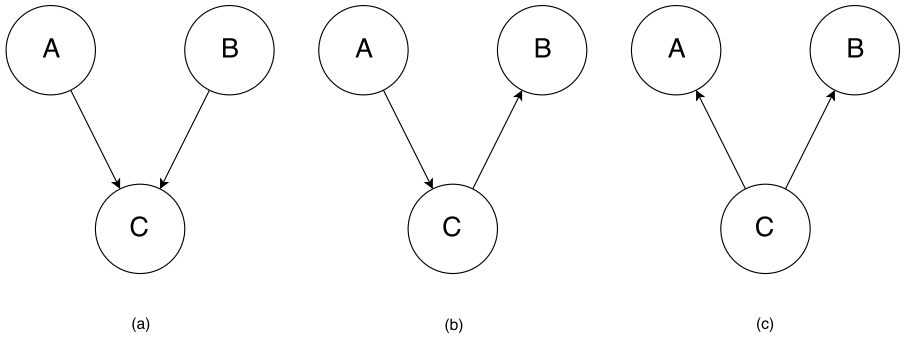



FIGURE 3.2: Examples of three types of relationship among three nodes (A-C-B).

The constraint-based method **symC** shown in Algorithm 3 and 4 is based on conditional independence tests. For an MB asymmetric pair, we perform the test conditioning on the union of the MBs of both nodes and they are not in the MB of each other if the test shows independence. If we happen to know the PC of both nodes, then we can check the set of common neighbors. No common neighbor means no common child, which implies that the MB pair is non-existent. We note that in the V-structure, conditioning on the common child $C$, the two parents, $A$ and $B$, are dependent, as shown in Figure 3.2 (a). On the other hand, for the networks in Figure 3.2 (b) and (c), conditioning on $C$, $A$ and $B$ are independent. If the set of common neighbors is not empty, we can perform tests conditioning on all subsets of them. Any dependency would indicate the existence of the MB pair.

For a PC asymmetric pair, we could apply a conditional independent test between $X$ and $T$ conditioned on all subsets of the intersection of their PCs if the intersection exists. However, the intersection is usually empty. Therefore for a particular pair $(X_i, X_j)$, we decide to test the conditional independence by conditioning on

**Algorithm 3: SymC($\mathcal{MB}$, $\mathcal{D}$)**

**Input:** The Markov blanket $\mathcal{MB}$ of each node obtained from a learning algorithm, the parent-child set $\mathcal{PC}$(optional), dataset $\mathcal{D}$

**Output:** symmetry corrected $\overline{\mathcal{MB}}$ of each node.

Extract all asymmetric MB pairs of nodes, $\mathcal{A}$ by going through all $\mathcal{MB}$ and checking if a pair of nodes is contained in the $\mathcal{MB}$ of each other.
The existent set $\mathcal{E} \leftarrow \emptyset$ and non-existent set $\mathcal{N} \leftarrow \emptyset$.
**for** each pair of nodes $(X_i, X_j)$ in $\mathcal{A}$ **do**
  **if** the $\mathcal{PC}$s of $X_i, X_j$ are not known **then**
    Apply a conditional independent test between $X_i$ and $X_j$ conditioned on the union of the $\mathcal{MB}(X_i)$ and $\mathcal{MB}(X_j)$.
    **if** the test shows dependence **then**
      The MB pair $(X_i, X_j)$ should exist and add it into $\mathcal{E}$.
    **else**
      The MB pair $(X_i, X_j)$ should not exist and add it into $\mathcal{N}$.
    **end if**
  **else**
    **if** the intersection of $\mathcal{PC}(X_i)$ and $\mathcal{PC}(X_j)$ is not empty **then**
      Apply conditional independent tests between $X_i$ and $X_j$ conditioned on all subsets of the intersection of their PCs.
      **if** any test shows dependence **then**
        The MB pair $(X_i, X_j)$ should exist and add it into $\mathcal{E}$.
      **else**
        The MB pair $(X_i, X_j)$ should not exist and add it into $\mathcal{N}$.
      **end if**
    **else**
      The MB pair $(X_i, X_j)$ should not exist and add it into $\mathcal{N}$.
    **end if**
  **end if**
**end for**
Update the $\mathcal{MB}$ of each node to symmetry corrected $\overline{\mathcal{MB}}$ according to $\mathcal{E}$ and $\mathcal{N}$.
**return** $\overline{\mathcal{MB}}$ of each node.

the all subsets of a set $\mathbf{S}$ to find a sep-set. If no sep-set $\mathbf{S}_{X_i,X_j}$ exists, which makes $X_i$ and $X_j$ independent, then it implies that there is an arc between $X_i$ and $X_j$ from Theorem 3.1.1. $\mathbf{S}$ can be the set of the rest of the nodes but it does not have to be that big. Since we have already learned the Markov blankets $\mathcal{MB}(X_i)$ and $\mathcal{MB}(X_j)$, (or $\mathcal{CPC}(X_i)$ and $\mathcal{CPC}(X_j)$), we can limit the size of $\mathbf{S}$ to reduce the number of tests. If $X_j \notin \mathcal{MB}(X_i)$, then by the definition of MB, $X_i$ is separated from $X_j$ by $\mathbf{S}_{X_i,X_j} = \mathcal{MB}(X_i)$. On the other hand, if $X_j \in \mathcal{MB}(X_i)$, then $\mathbf{S}_{X_i,X_j} \subseteq \mathcal{MB}(X_i) \setminus X_j$ and $\mathbf{S}_{X_i,X_j} \subseteq \mathcal{MB}(X_j) \setminus X_i$ [30]. To further reduce the

**Algorithm 4: SymC**($\mathcal{PC}$, $\mathcal{MB}$, $\mathcal{D}$)

**Input:** The parent-child set $\mathcal{PC}$ and the Markov blanket (or candidate parent-child set) $\mathcal{MB}$ of each node obtained from a learning algorithm, dataset $\mathcal{D}$

**Output:** symmetry corrected $\overline{\mathcal{PC}}$ of each node.

Extract all asymmetric PC pairs of nodes $\mathcal{A}$ by going through all $\mathcal{PC}$s and checking if a pair of nodes is contained in $\mathcal{PC}$ of each other.
The existent set $\mathcal{E} \leftarrow \emptyset$ and non-existent set $\mathcal{N} \leftarrow \emptyset$.
**for** each pair of nodes $(X_i, X_j)$ in $\mathcal{A}$ **do**
    Set $\mathcal{S}$ to be the smaller one of $\{\mathcal{MB}(X_i), \mathcal{MB}(X_j)\}$.
    **if** $\mathcal{S}$ is empty **then**
        $X_i$ and $X_j$ should be not connected and add them into set $\mathcal{N}$.
    **else**
        Apply conditional tests on $X_i$ and $X_j$ conditioned on all subsets of $\mathcal{S}$.
        **if** all tests show dependence **then**
            $X_i$ and $X_j$ should be connected and add them into $\mathcal{E}$.
        **else**
            $X_i$ and $X_j$ should be not connected and add them into $\mathcal{N}$.
        **end if**
    **end if**
**end for**
Update the $\mathcal{PC}$ of each node to symmetry corrected $\overline{\mathcal{PC}}$ according to $\mathcal{E}$ and $\mathcal{N}$.
**return** $\overline{\mathcal{PC}}$ of each node

number of tests, we take the smaller MB from $\mathcal{MB}(X_i)$ and $\mathcal{MB}(X_j)$ as $\mathbf{S}$. For example, in Figure 3.1, for the asymmetric PC pair $(C, F)$, let us assume $\mathcal{MB}(C) = \mathcal{PC}(C) = \{A, D, F\}$ and $\mathcal{MB}(F) = \mathcal{PC}(F) = \{E, H\}$, as $|\mathcal{MB}(C)| > |\mathcal{MB}(F)|$, we will test all subsets of $\{E, H\}$.

### 3.3.4 Time and Space Complexity

In Algorithm 2, 3 and 4, the for loop is executed at most $n(n-1)/2$ times, i.e., the number of possible pairs of nodes, where $n$ is the number of nodes. Within one loop, for **symG**, the cost is dominated by the greedy search and for **symC**, the cost mainly depends on the size of the conditioning set.

For **symG**, the number of nodes in the search depends on the size of the PCs and MBs of the two nodes in the asymmetric pair. For PC, it is approximately limited to twice the maximum degree of the network, $m$, as the learned PCs are not exact.

For MB, in the worst case, it may contain all nodes as the MB is not limited by the maximum degree. In practice, however, the number of asymmetric pairs and the number of nodes in the greedy search are actually much lower.

For **symC**, the number of tests we need to perform is $2^b$, where $b$ is the size of the conditioning set, which is bounded by the size of MB. Though still in the worst case, the MB may cover all nodes, in our experiments, it is not very large.

## 3.4 Empirical Evaluation

TABLE 3.1: Information about the networks used in the experiments.

| Name | Nodes | Arcs | Parameters | Maximum in-degree |
|---|---|---|---|---|
| ALARM | 37 | 46 | 509 | 4 |
| ANDES | 223 | 338 | 1157 | 6 |
| ASIA | 8 | 8 | 18 | 2 |
| BARLEY | 48 | 84 | 114005 | 4 |
| CHILD | 20 | 25 | 230 | 2 |
| DIABETES | 413 | 602 | 429409 | 2 |
| HAILFINDER | 56 | 66 | 2656 | 4 |
| LINK | 724 | 1125 | 14211 | 3 |
| MILDEW | 35 | 46 | 540150 | 3 |
| HEPAR2 | 70 | 123 | 1453 | 6 |
| INSURANCE | 27 | 52 | 984 | 3 |
| MUNIN1 | 186 | 273 | 15622 | 3 |
| PATHFINDER | 135 | 200 | 77155 | 5 |
| PIGS | 441 | 592 | 5618 | 2 |
| WATER | 32 | 66 | 10083 | 5 |
| WIN95PTS | 76 | 112 | 574 | 7 |

### 3.4.1 Experimental Setup

We ran our experiments using software R [38] on Intel Xeon E5-1650 Processor(Six Core HT, 3.2GHz Turbo) with 16GB of RAM. We evaluated our two proposed methods using data generated from probability distributions of 16 BNs shown in Table 3.1, from the BN repository of **bnlearn** [1] [13]. The GS, IAMB, MMPC, and SI-HITON-PC structure learning algorithms were implemented using the R package **bnlearn** [13]. The tests used were the default tests based on the asymptotic mutual information with $\alpha = 0.05$. The default maximum allowed the size of the conditioning set was unlimited. The SLL algorithm was implemented using the codes from [27]. Our methods modified the intermediate steps of the algorithms

[1] http://www.bnlearn.com/bnrepository/

to perform the symmetry correction. We generated 500, 1,000 and 5,000 samples from each BNs with 10 different seeds. Then we applied each algorithm to these 10 sets of data to learn the PC and MB. After that, we applied the AND-rule, and our methods, **symG** and **symC** to the learned PC and MB to compute the results and took the averages. We used the R package **pcalg** [39] to get the extension of PDAG and **bnlearn** [13] to compute the *Structural Hamming Distance* (SHD) [20], Bayesian information criteria (BIC) [17] and *Bayesian Dirichlet equivalent uniform* (BDeu) [18]. The imaginary sample size of BDeu was set to 1. The significance of the results was tested by paired t-test with $\alpha = 0.05$ and shown in bold (if they are better than the baseline) and underline (if they are worse than the baseline). In addition, we ran some global score-based methods, Fast Greedy Equivalence Search (FGES) [40], a faster version of the Greedy Equivalence Search (GES) [19], from package **r-causal**, which is the R Wrapper for Tetrad Library [2], Hill Climbing (HC) search in DAG with arc operations from **bnlearn** and fast constrained Hill Climbing (FastCHC) in default settings [3] [34].

In the three stages of the constraint-based algorithms [30], the optional learning of Markov blankets makes a difference, so we applied our proposed symmetry correction methods accordingly to the GS and IAMB algorithms as shown in Algorithm 1. These algorithms find the MB first and then perform the symmetry correction for MB. They find the PC using the MB and perform the symmetry correction for PC. Hence, the symmetry corrections are done twice. We only listed the MB results since only a few asymmetric pairs of PC were obtained from the symmetry corrected MB. MMPC and SI-HITON-PC algorithms find the candidate PC (CPC) first and use it to find the PC. Then a symmetry correction method is applied to the PC. The corrected PC can be used to find the MB or it simply produces the fake MB. The fake MB is then used in the detection of V-structures to obtain PDAG. SLL is similar to MMPC and SI-HITON-PC but it outputs both the PC and MB. A symmetry correction method is first applied to the PC and then the MB. As the corrected PC was known when correcting the MB using **symC**, we used the information of the corrected PC as shown in Algorithm 3. The average asymmetric pairs were shown in Table 3.2. Some networks were omitted for SLL because the dynamic programming applied in SLL had a high cost for handling

[2] http://www.phil.cmu.edu/tetrad/

[3] http://simd.albacete.org/supplements/FastCHC.html

large networks. Also, we ran out of memory for DIABETES, MUNIN1, and PIGS when computing the scores.

TABLE 3.2: The average number of asymmetric MB pairs for GS, IAMB and asymmetric PC pairs for MMPC, SI-HITON-PC, and SLL with 1,000 samples. Some networks are omitted for SLL ("-") due to the high computation cost.

| | MB | | PC | | |
|---|---|---|---|---|---|
| | GS | IAMB | MMPC | SI-HITON-PC | SLL |
| ALARM | 34.7 | 25.4 | 25.2 | 20.6 | 7 |
| ANDES | 373.5 | 477.8 | 160.3 | 204.9 | 128.2 |
| ASIA | 6.4 | 5.1 | 4.8 | 3.4 | 0.6 |
| BARLEY | 62 | 32.9 | 43.9 | 26.6 | 19.6 |
| CHILD | 27 | 14.7 | 11.2 | 9.8 | 1.2 |
| DIABETES | 449.5 | 200.3 | 372.9 | 176 | - |
| HAILFINDER | 79.7 | 50.5 | 38.3 | 37.4 | 21.3 |
| LINK | 1532.4 | 1642.3 | 580.2 | 744.2 | - |
| MILDEW | 22.5 | 20.3 | 20.8 | 16.5 | - |
| HEPAR2 | 112.2 | 122.4 | 56.4 | 58.1 | 12.2 |
| INSURANCE | 24.9 | 24.5 | 19.9 | 18 | 7.8 |
| MUNIN1 | 207.3 | 127.9 | 133 | 113.5 | - |
| PATHFINDER | 104 | 125.8 | 99.8 | 106 | - |
| PIGS | 1214.5 | 233 | 361.3 | 244.3 | - |
| WATER | 5.4 | 6.2 | 4.9 | 5.3 | 12.6 |
| WIN95PTS | 87.9 | 92.5 | 47 | 50.1 | 77.6 |

### 3.4.2 Performance Measures

We compared our two proposed symmetry correction methods with the AND-rule over four constraint-based local BN structure learning algorithms (GS, IAMB, MMPC, and SI-HITON-PC) and one score-based algorithm, SLL. We measured their performance based on three criteria. The first one is the set difference in PC and MB between the learned network and the true network and the second one is the SHD between the learned network and the true network. The third one is the BIC and BDeu scores of training data on the learned BN.

Let $\overline{\mathcal{LS}}_{\mathcal{SC}}$ be the symmetry corrected local structures (PCs or MBs) using symmetry correction method $\mathcal{SC}$ and $\mathcal{LS}_{true}$ be the local structures of the true network. The set difference in PC(MB), $\text{Diff}_{\mathcal{SC}}$, is measured by the symmetrical difference between the true PC(MB) and the learned PC(MB). Let $a$ be the number of asymmetric pairs and $c_{\mathcal{SC}}$ be the number of wrongly corrected pairs using $\mathcal{SC}$. $\mathcal{SC}$ can be **symG** or **symC**.

$$\begin{aligned}\text{Diff}_{\mathcal{SC}} &= \text{Diff}(\mathcal{LS}_{\mathcal{SC}}, \mathcal{LS}_{true}) \\ &= \frac{1}{2a}\sum_{X_i \in V} |\mathcal{LS}_{\mathcal{SC}}(X_i) \cup \mathcal{LS}_{true}(X_i)| - |\mathcal{LS}_{\mathcal{SC}}(X_i) \cap \mathcal{LS}_{true}(X_i)| \\ &= \frac{c_{\mathcal{SC}}}{a}\end{aligned} \tag{3.1}$$

It is summed over all variables and normalized with twice the number of asymmetric pairs as one pair affects two PC(MB).

$$D_{\mathcal{SC}} = \text{Diff}_{\text{AND-rule}} - \text{Diff}_{\mathcal{SC}} = \frac{c_{\text{AND-rule}} - c_{\mathcal{SC}}}{a} \tag{3.2}$$

$D_{\mathcal{SC}}$ measures how close the learned PC(MB) is to the true network, compared to the AND-rule method. The more positive $D_{\mathcal{SC}}$ is, the better the performance of our compared method is. As the only difference in the results of symmetry correction methods is produced on the asymmetric pairs, it can also be viewed as the difference in the number of asymmetric pairs corrected correctly between our method and the baseline method as shown in 3.2. For example, assume there are 10 asymmetric pairs, the baseline method gives 2 right corrections and 8 wrong corrections while our method gives 4 right corrections and 6 wrong corrections, then the output is $(8-6)/10 = 0.2$. It is worth noting that the output is bounded between -1 and 1, as the extreme cases happen when one method gives all right corrections while the other gives all wrong corrections.

To compute the BIC or BDeu scores, we need to obtain the DAG extension of the PDAG. We orient the edges in the learned skeleton using the detected V-structures and the Meek's rule to get the PDAG, which may contain undirected edges and then extend this PDAG to DAG. However, not all PDAG can be extended to DAG. A successful extension means the undirected edges can be directed without creating any additional V-structure or directed cycle. Finally, the DAG is fitted with data to get the final BN and its corresponding BIC or BDeu score. A higher score indicates a better model. On the other hand, a better symmetry corrected MB and PC can produce a better skeleton. This can be used as the restriction phase of the hybrid algorithms, which applies the hill-climbing algorithm on the skeleton. Towards this end, we compute the SHD for the PDAG and the DAG from the hybrid algorithms and the BIC and BDeu scores for the DAG extension of the PDAG and the DAG from the hybrid algorithms.

TABLE 3.3: The summary of the results **symC**("C") and **symG**("G") for GS, IAMB, MMPC, and SI-HITON-PC in total. The numbers $a/b$ represents $a$ significantly better results compared to the AND-rule baseline and $b$ significantly worse results. Across 16 networks and 4 algorithms, the maximum number of significant results is 64.

| Sample size | 500 | | 1000 | | 5000 | |
|---|---|---|---|---|---|---|
| | C | G | C | G | C | G |
| diff in MB/PC | 29/14 | 50/13 | 33/14 | 52/10 | 35/15 | 50/11 |
| SHD | 10/23 | 28/22 | 18/20 | 33/21 | 20/25 | 37/21 |
| SHD hybrid | 21/13 | 28/19 | 25/12 | 33/19 | 34/13 | 42/15 |
| BIC | 17/2 | 25/3 | 18/2 | 23/3 | 20/3 | 19/5 |
| BIC hybrid | 45/0 | 64/0 | 48/0 | 63/1 | 54/0 | 64/0 |
| BDeu | 28/1 | 35/1 | 24/0 | 31/1 | 24/0 | 27/0 |
| BDeu hybrid | 46/0 | 64/0 | 48/0 | 63/1 | 55/0 | 64/0 |

### 3.4.3 Comparison Results

Detailed comparison results are presented in Table 3.4-3.7,3.9-3.17. Table 3.3 summarizes these experimental results in terms of the number of significantly better results and significantly worse results. The hybrid algorithms using our **symG** and **symC** show a big improvement over the AND-rule. This is because better learned skeleton gives the hybrid algorithm a better search space.

**Set Differences in PC and MB** The comparison results in Table 3.4 and 3.5 show the average difference $D_{symG}$ between the set difference $\text{Diff}_{symG}$ using our score-based method, **symG** and $\text{Diff}_{\text{AND-rule}}$ using the AND-rule baseline method on the 16 datasets described in Table 3.1. The comparison results in Table 3.6 and 3.7 show the average difference $D_{symC}$ between set difference $\text{Diff}_{symC}$ using our constraint-based method, **symC** and $\text{Diff}_{\text{AND-rule}}$. Since the numbers in Table 3.4 and 3.5 are mostly positive, it is clear that **symG** performs better than the AND-rule. From Table 3.6 and 3.7, one observes that the improvement of **symC** is not as significant as **symG**. The results for the MB of MMPC and SI-HITON-PC are not shown here because the skeleton could be constructed from PC directly. One notes that the fake MB can provide all V-structure candidates for V-structure detection.

**Structural Hamming Distance** From Table 3.3 and 3.9-3.11, one observes that **symG** generally performed better than **symC**. As the PDAG is obtained from

TABLE 3.4: The average difference $D_{symG}$ between set difference $\text{Diff}_{symG}$ in MB of constraint-based algorithms (GS and IAMB) using our score-based method **symG** and $\text{Diff}_{\text{AND-rule}}$ of the AND-rule baseline method normalized with twice the number of asymmetric pairs.

| | GS | | | IAMB | | |
|---|---|---|---|---|---|---|
| Sample size | 500 | 1000 | 5000 | 500 | 1000 | 5000 |
| ALARM | **0.09** | **0.06** | -0.03 | **0.58** | **0.66** | **0.51** |
| ANDES | **0.18** | **0.27** | **0.42** | **0.08** | **0.17** | **0.24** |
| ASIA | -0.06 | -0.17 | -0.23 | **0.6** | **0.7** | **0.82** |
| BARLEY | **0.12** | **0.02** | **0.1** | **0.22** | **0.27** | **0.38** |
| CHILD | **0.13** | 0.07 | **0.2** | **0.31** | **0.25** | **0.49** |
| DIABETES | **0.11** | **0.05** | -0.12 | **0.37** | **0.14** | **0.19** |
| HAILFINDER | **0.2** | **0.27** | **0.18** | **0.37** | **0.36** | **0.32** |
| LINK | -0.08 | -0.07 | -0.06 | **0.05** | **0.08** | **0.18** |
| MILDEW | **0.1** | **0.25** | **0.32** | **0.29** | **0.41** | **0.28** |
| HEPAR2 | **0.08** | **0.15** | **0.32** | **0.06** | **0.13** | **0.21** |
| INSURANCE | **0.35** | **0.44** | **0.42** | **0.47** | **0.56** | **0.73** |
| MUNIN1 | -0.29 | -0.32 | -0.34 | **0.19** | **0.26** | **0.08** |
| PATHFINDER | -0.05 | **0.07** | **0.33** | -0.21 | -0.24 | **0.11** |
| PIGS | -0.01 | -0.01 | -0.06 | **0.97** | **0.99** | **0.99** |
| WATER | **0.51** | **0.23** | **0.22** | **0.49** | **0.36** | **0.44** |
| WIN95PTS | **0.44** | **0.5** | **0.44** | **0.52** | **0.63** | **0.75** |

detecting and applying V-structures to the skeleton, the SHD partially depends on the V-structure detection. Hence, it may not fully reflect the improvements from set differences shown in Table 3.4-3.7.

### BIC and BDeu Scores

For Table 3.12-3.17, the results of the PDAG are included. PDAG may be extended to DAG and the average scores are computed. The one that cannot be extended to DAG in all 10 runs is labelled with "n.e.". As the extension requires no additional V-structure, the PDAG sometimes extended to a DAG with a low score. On the other hand, the DAG from hybrid algorithms has shown great improvement over the AND-rule in both BIC and BDeu scores.

### SLL Results

One observes from Table 3.8 that the score-based local algorithm has limited improvement against the AND-rule. However, **symC** is clearly more suitable than **symG** for SLL. Results in Table 3.18 for PDAG are mostly negative values. Also,

TABLE 3.5: The average difference $D_{symG}$ between set difference $\text{Diff}_{symG}$ in PC of constraint-based algorithms (MMPC and SI-HITON-PC) using our score-based method **symG** and $\text{Diff}_{\text{AND-rule}}$ of the AND-rule baseline method normalized with twice the number of asymmetric pairs. "*" indicates the test fails due to a constant difference for all 10 iterations.

| | MMPC | | | SI-HITON-PC | | |
|---|---|---|---|---|---|---|
| Sample size | 500 | 1000 | 5000 | 500 | 1000 | 5000 |
| ALARM | **0.3** | **0.44** | **0.39** | **0.39** | **0.41** | **0.38** |
| ANDES | -0.01 | **0.08** | **0.23** | -0.26 | -0.12 | **0.03** |
| ASIA | -0.17 | -0.33 | -0.59 | **0.53** | **0.64** | 0.68* |
| BARLEY | **0.1** | **0.04** | **0.19** | **0.23** | **0.3** | **0.41** |
| CHILD | **0.6** | **0.49** | **0.88** | **0.47** | **0.46** | **0.6** |
| DIABETES | **0.11** | 0.01 | **0.03** | **0.39** | **0.22** | -0.01 |
| HAILFINDER | **0.49** | **0.58** | **0.46** | **0.43** | **0.4** | **0.45** |
| LINK | -0.07 | **0.11** | -0.03 | -0.18 | -0.17 | -0.13 |
| MILDEW | **0.12** | **0.27** | **0.28** | **0.33** | **0.48** | 0.01 |
| HEPAR2 | **0.12** | **0.27** | **0.47** | **0.1** | **0.2** | **0.38** |
| INSURANCE | **0.42** | **0.51** | **0.56** | **0.46** | **0.35** | **0.46** |
| MUNIN1 | -0.13 | -0.03 | -0.05 | **0.24** | **0.26** | -0.13 |
| PATHFINDER | -0.04 | **0.08** | **0.23** | -0.18 | -0.22 | **0.29** |
| PIGS | **0.86** | **0.94** | **0.91** | **0.7** | **0.6** | **0.51** |
| WATER | **0.74** | **0.53** | **0.5** | **0.54** | **0.33** | **0.31** |
| WIN95PTS | **0.42** | **0.38** | **0.31** | **0.29** | **0.28** | **0.26** |

in Table 3.19 and 3.20, there are only several positive values. This could be due to poor V-structure detections or restricted DAG extension. For DAGs returned from the hybrid algorithm (SLL+ hill-climbing), the results are quite positive. However, those improvements are much less compared to those from the constraint-based algorithms in Table 3.12-3.17. This may be due to the fact that SLL produces less asymmetric pairs than the constraint-based algorithms except for WATER and WIN95PTS (see Table 3.2).

### 3.4.4 Discussions

The performance of FGES and HC in Table 3.9-3.17 are better than the constraint-based algorithms. However, the computational time for these score-based algorithms is usually relatively high. For example, for the network ANDES, with 1,000 samples, MMPC spends 2.2 seconds but HC spends 5.0 seconds and FGES spends 30.9 seconds. There are a few cases such as HAILFINDER using SI-HITON-PC

TABLE 3.6: The average difference $D_{symC}$ between set difference $\text{Diff}_{symC}$ in MB of constraint-based algorithms (GS and IAMB) using our constraint-based method **symC** and $\text{Diff}_{\text{AND-rule}}$ of the AND-rule baseline method normalized with twice the number of asymmetric pairs. "*" indicates the test fails due to a constant difference for all 10 iterations.

| | GS | | | IAMB | | |
|---|---|---|---|---|---|---|
| Sample size | 500 | 1000 | 5000 | 500 | 1000 | 5000 |
| ALARM | 0.039 | **0.034** | 0.041 | 0 | 0.004 | **0.023** |
| ANDES | **0.196** | **0.255** | **0.409** | **0.004** | **0.003** | 0.001 |
| ASIA | 0.003 | 0 | 0 | 0 | -0.032 | 0 |
| BARLEY | 0.004 | 0.003 | **0.038** | 0 | 0 | **0.026** |
| CHILD | **0.094** | **0.101** | **0.23** | -0.008 | -0.007 | -0.018 |
| DIABETES | **0.055** | -0.04 | -0.079 | 0 | 0.005 | **0.094** |
| HAILFINDER | **0.122** | **0.177** | **0.139** | 0 | 0 | 0.006 |
| LINK | -0.017 | -0.015 | **0.035** | **0.001** | **0.005** | **0.035** |
| MILDEW | **0.058** | **0.165** | **0.232** | 0 | 0 | 0.149* |
| HEPAR2 | **0.028** | **0.049** | **0.109** | -0.003 | -0.003 | -0.001 |
| INSURANCE | **0.119** | **0.227** | **0.282** | 0 | 0 | 0.004 |
| MUNIN1 | -0.048 | -0.04 | -0.044 | 0 | 0.002 | **0.053** |
| PATHFINDER | 0.007 | **0.039** | **0.065** | 0 | 0 | -0.008 |
| PIGS | 0.001 | **0.035** | -0.001 | 0.001 | 0 | 0 |
| WATER | **0.225** | **0.041** | **0.092** | 0 | 0 | 0 |
| WIN95PTS | **0.159** | **0.151** | **0.198** | **0.028** | **0.041** | **0.044** |

(see Table 3.9), which outperforms FGES and HC with the help of our **symG** method.

The score-based and constraint-based algorithms can be viewed as two different ways to capture the structure information about BN from data. To understand the relationship between the type of algorithm and the type of symmetry correction method, we investigate the four kinds of combinations as follows.

- constraint-based algorithm + **symG**
- constraint-based algorithm + **symC**
- score-based algorithm + **symG**
- score-based algorithm + **symC**

The empirical results of average differences in PC and MB set differences show that the most improvement can be obtained in the case of constraint-based algorithm + **symG**. For constraint-based algorithms, **symG** is better than **symC**.

TABLE 3.7: The average difference $D_{symC}$ between set difference $\text{Diff}_{symC}$ in PC of constraint-based algorithms (MMPC and SI-HITON-PC) using our constraint-based method **symC** and $\text{Diff}_{\text{AND-rule}}$ of the AND-rule baseline method normalized with twice the number of asymmetric pairs. "*" indicates the test fails due to a constant difference for all 10 iterations.

| | MMPC | | | SI-HITON-PC | | |
|---|---|---|---|---|---|---|
| Sample size | 500 | 1000 | 5000 | 500 | 1000 | 5000 |
| ALARM | -0.04 | 0.03 | **0.16** | **0.29** | **0.17** | **0.16** |
| ANDES | -0.17 | -0.18 | -0.17 | -0.37 | -0.37 | -0.43 |
| ASIA | -0.05 | -0.03 | -0.06 | **0.16** | **0.21** | 0.31* |
| BARLEY | -0.24 | -0.41 | -0.02 | **0.17** | **0.3** | **0.14** |
| CHILD | **0.32** | **0.31** | **0.7** | **0.5** | **0.3** | **0.28** |
| DIABETES | -0.16 | -0.41 | -0.2 | **0.12** | -0.02 | -0.45 |
| HAILFINDER | **0.25** | **0.28** | **0.21** | -0.06 | -0.05 | -0.08 |
| LINK | -0.24 | -0.05 | -0.29 | -0.22 | -0.29 | -0.28 |
| MILDEW | **0.02** | **0.16** | **0.17** | **0.02** | **0.09** | -0.24 |
| HEPAR2 | -0.05 | **0.15** | **0.39** | -0.06 | **0.05** | **0.23** |
| INSURANCE | **0.28** | **0.4** | **0.33** | **0.21** | **0.23** | **0.24** |
| MUNIN1 | -0.27 | -0.16 | -0.2 | **0.02** | **0.1** | -0.26 |
| PATHFINDER | -0.01 | **0.2** | **0.34** | -0.18 | -0.2 | **0.27** |
| PIGS | **0.74** | **0.77** | **0.81** | **0.45** | **0.25** | **0.13** |
| WATER | **0.41** | **0.48** | **0.54** | **0.15** | **0.34** | **0.27** |
| WIN95PTS | **0.26** | **0.18** | **0.09** | -0.07 | -0.08 | -0.17 |

For the score-based algorithm, SLL, **symC** is better than **symG**. To some extent, this coincides with our intuition that the combination of different types of structure learning algorithm and symmetry correction method outperforms that of the same type of learning algorithm and symmetry correction method. This can be explained by the weaknesses of the structure learning algorithms, low power hypothesis testing and sub-optimal points in the search discussed in Section 3.3.1. A combination of a learning algorithm and a different type symmetry correction method can alleviate the weakness of the learning algorithm to achieve better overall performance. This hybrid framework approach can be understood as structure learning using two different types of BN structure information compared to using only information used in either the constraint-based algorithm or the score-based algorithm.

Lastly, to illustrate the complexity of the proposed **symG** and **symC** methods, we listed the maximum number of nodes in the greedy search and the maximum

TABLE 3.8: The average difference $D_{symC}$ ($D_{symG}$) between set difference $\text{Diff}_{symC}$ ($\text{Diff}_{symG}$) in PC of score-based algorithm SLL using **symC** (**symG**), and the AND-rule baseline method normalized with twice the number of asymmetric pairs. "*" indicates the test fails due to a constant difference for all 10 iterations.

| | **symC** | | | **symG** | | |
|---|---|---|---|---|---|---|
| Sample size | 500 | 1000 | 5000 | 500 | 1000 | 5000 |
| ALARM | 0.04 | -0.02 | -0.06 | -0.14 | -0.12 | -0.29 |
| ANDES | 0 | -0.01 | 0 | -0.15 | -0.13 | -0.09 |
| ASIA | -0.07 | 0.08 | -0.25 | -0.27 | 0.25 | 0 |
| BARLEY | **0.05** | **0.09** | **0.15** | -0.09 | 0.04 | **0.14** |
| CHILD | -0.36 | -0.05 | **0.6** | -0.5 | 0.11 | 0.45* |
| HAILFINDER | 0 | 0 | 0 | -0.35 | -0.48 | -0.61 |
| HEPAR2 | -0.06 | -0.06 | **0.28** | -0.07 | -0.02 | **0.13** |
| INSURANCE | **0.12** | -0.04 | **0.18** | 0.02 | **0.1** | **0.5** |
| WATER | 0 | 0.01 | **0.1** | 0 | 0 | 0.01 |
| WIN95PTS | 0 | 0.02 | 0.01 | -0.05 | -0.06 | -0.12 |

size of conditioning set used in our experiments for each dataset in Table 3.21. We also included the running time of **symG** and **symC** in Table 3.22.

## 3.5 Summary

We investigated the symmetry correction problem which exists in the BN local structure learning algorithms and proposed a score-based symmetry correction method **symG** and a constraint-based method **symC**. We found that there is a big improvement when **symG** is applied to the constraint-based algorithms. From our experimental results, we found that for constraint-based learning algorithms, **symG** is better than **symC**, and **symC** has reasonable performance. For the score-based learning algorithm, **symC** results in a limited improvement compared to the baseline, but the use of **symG** does not result in better performance. This coincides with our intuition that a hybrid algorithm that combines different types of learning algorithms and symmetry correction methods is better than a combination of a learning algorithm and a symmetry correction method of the same type.

TABLE 3.9: The average SHD of PDAG for GS, IAMB, MMPC, and SI-HITON-PC and DAG for the corresponding hybrid algorithms, using the AND-rule("A"), **symC**("C") and **symG**("G") with 500 samples.

| | PDAG | | | | | | | | | | | | DAG | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GS | | | IAMB | | | MMPC | | | SI-HITON-PC | | | GS | | | IAMB | | | MMPC | | | SI-HITON-PC | | | FGES | HC | FastCHC |
| | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | | | |
| ALARM | 46.1 | 48.8 | 48.6 | 39.2 | 39.2 | **29.8** | 41.9 | 47.9 | **37.3** | 39 | 37.8 | **36.1** | 47.2 | 47.5 | 47.3 | 38.7 | 38.7 | **29.9** | 41.5 | 43.3 | **36.9** | 39.3 | **33** | **33.2** | 15.2 | 33.6 | 77.6 |
| ANDES | 279.1 | **218.5** | **227.2** | 238.1 | 236.9 | **227.5** | 239.5 | 252.9 | **229.2** | 223.8 | 294.1 | 277.9 | 274.9 | **205.3** | **210.6** | 225.2 | **223.7** | **203.2** | 227.3 | **210.7** | **198** | 211.3 | 220.2 | 210.1 | 165.2 | 208.5 | 699.7 |
| ASIA | 5.4 | 5.4 | 7.7 | 5.4 | 5.4 | **2.7** | 5.1 | 5.2 | 6 | 6 | **4.7** | **2.9** | 5.1 | 5.1 | 5.3 | 5.1 | 5.1 | **2.3** | 4.8 | 4.9 | 5.3 | 5.7 | **4.1** | **2.4** | 2.7 | 2.3 | 12.1 |
| BARLEY | 83.4 | 83.3 | 86.3 | 82 | 82 | **81.3** | 83.3 | 99.6 | 85.9 | 79.1 | 80.9 | 80.6 | 82.8 | 82.7 | 85.8 | 84 | 84 | 84.1 | 82.8 | 85.8 | 85.5 | 84 | 84 | 85 | 78.4 | 86.5 | 87.6 |
| CHILD | 23.6 | 23 | **19.3** | 16.1 | 16.1 | 16.9 | 21.3 | **19.8** | **16.2** | 17.8 | **15.8** | 16.8 | 23.7 | 23.1 | 23 | 18.6 | 18.6 | **16.2** | 21.4 | **19.5** | **17.7** | 18.2 | **17** | **16.4** | 12.5 | 16.6 | 23.3 |
| DIABETES | 601.1 | 600.3 | 630.6 | 625.7 | 625.7 | **624** | 601.4 | 721 | 601.9 | 625.4 | 633.5 | **620.5** | 601.4 | **597.5** | 632.2 | 625 | 625 | 626.6 | 601.7 | 635.1 | 634 | 625.8 | 633.8 | 634.6 | 653.2 | 691.9 | 663.2 |
| HAILFINDER | 62.2 | **56.3** | **49.3** | 55.2 | 55.2 | **35** | 60.4 | **56.4** | **39.7** | 51.7 | 56.9 | **36.8** | 63.8 | **58.1** | **57.8** | 58 | 58 | **44.4** | 62.6 | **53.3** | **48.5** | 58.2 | **56.1** | **44.4** | 57.4 | 52.2 | 58.6 |
| LINK | 1079.6 | 1111.5 | 1245.1 | 1039.9 | 1041.7 | 1165.4 | 1033 | 1282.2 | 1199.5 | 1088.2 | 1375.4 | 1375.4 | 1080.9 | 1105.1 | 1221.8 | 1052.8 | 1053.6 | 1067.5 | 1034.7 | 1162.3 | 1143.6 | 1086.1 | 1084.7 | 1208.1 | 882.4 | 1351.3 | 2668.6 |
| MILDEW | 46 | 46 | 46.1 | 45.9 | 45.9 | 45.7 | 46.1 | 46.7 | 46.2 | 46 | 45.7 | 45.8 | 46 | 46 | 46.1 | 46 | 46 | 46 | 46 | 46.1 | 46.1 | 46 | 46 | 46 | 49.8 | 47.5 | 46.9 |
| HEPAR2 | 118.4 | 117.5 | 117.2 | 121.1 | 121.5 | **113** | 122.1 | 127.1 | 120.3 | 124.8 | 125.7 | **116.4** | 116.2 | **114.4** | **111.9** | 117.9 | 117.7 | **110.8** | 118.5 | **111.9** | **110.6** | 120.3 | **106.7** | **106.4** | 104.5 | 106.4 | 234.0 |
| INSURANCE | 46.9 | 46.4 | 46.6 | 47.9 | 48.1 | 46.2 | 47 | 49 | **44.1** | 49.3 | 46.8 | **43.8** | 47.4 | **45.9** | **45.2** | 45 | 45 | 45.2 | 47.4 | 47.6 | **44.3** | 46 | 44.9 | 44.6 | 35 | 47.1 | 62.1 |
| MUNIN1 | 275.6 | 286.9 | 325.9 | 279 | 279 | 290.5 | 275.9 | 321.2 | 305.4 | 277.5 | 284.7 | 293.3 | 275 | 286 | 327.5 | 277.1 | 277.1 | 288.8 | 275.2 | 311.8 | 307.7 | 277 | 279.7 | 280.2 | 219.4 | 286.7 | 583.5 |
| PATHFINDER | 192.2 | 192.8 | 202.2 | 194.2 | 194.2 | 228.4 | 193 | 198.6 | 201.2 | 195 | 210.1 | 225.7 | 193.1 | 193.8 | 200.7 | 192.1 | 192.1 | 218.6 | 193.3 | 196.8 | 200.4 | 193.5 | 204.5 | 215.4 | 251.7 | 241.8 | 454.3 |
| PIGS | 585 | 586 | 618.9 | 343.6 | 343.4 | **195.7** | 509.1 | **231.3** | **201.1** | 396.6 | **263.2** | **168.2** | 585.3 | 584.3 | 616.5 | 379.7 | 379.5 | **119.9** | 519.4 | **195.4** | **171.8** | 409.7 | **99.5** | **79.3** | 0.4 | 130.2 | 512.0 |
| WATER | 60.5 | 60.5 | 60.4 | 60.4 | 60.4 | 60.4 | 60.4 | 60.4 | 60.2 | 60.3 | 60.2 | 60.3 | 60.1 | 60.1 | 60 | 60 | 60 | 60 | 60.2 | 60 | 60 | 60 | 60 | 60 | 59.9 | 60.2 | 98.7 |
| WIN95PTS | 98.2 | **94.2** | **87.5** | 102.2 | 101.9 | **90.8** | 99.7 | **93.4** | **82.8** | 99 | 104.1 | **84.7** | 97.7 | **90.2** | **80.9** | 99.2 | **98.7** | **90.9** | 99.9 | **84.9** | **79.7** | 97.8 | **96** | **84** | 70.6 | 73.6 | 385.8 |

TABLE 3.10: The average SHD of PDAG for GS, IAMB, MMPC, and SI-HITON-PC and DAG for the corresponding hybrid algorithms, using the AND-rule("A"), **symC**("C") and **symG**("G") with 1,000 samples.

| | PDAG | | | | | | | | | | | | DAG | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GS | | | IAMB | | | MMPC | | | SI-HITON-PC | | | GS | | | IAMB | | | MMPC | | | SI-HITON-PC | | | FGES | HC | FastCHC |
| | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | | | |
| ALARM | 47.1 | 48.8 | 48.3 | 34.3 | 34.1 | **24.6** | 39.6 | 42.1 | **32.2** | 33.1 | 35.5 | 31.5 | 47.6 | 48.4 | **44.4** | 31.1 | 30.9 | **20.5** | 37.6 | 37.6 | **27.8** | 30.7 | **28.3** | **24.5** | 11.2 | 27 | 40.8 |
| ANDES | 264.9 | **186.8** | **184.4** | 203.5 | **202.2** | **176** | 204 | 220.4 | **178.3** | 190.7 | 260.2 | 212.4 | 260.3 | **173.8** | **166.5** | 193.1 | **191.6** | **161.3** | 193.1 | **170** | **155.7** | 178.7 | 178.8 | **159.4** | 125.8 | 155.4 | 214.3 |
| ASIA | 5.5 | 5.5 | 7.7 | 5.5 | 5.7 | **2** | 5.1 | 5 | 5.5 | 5.1 | **3.3** | **1.5** | 5 | 5 | 5.4 | 5 | 5.1 | **1.5** | 4.6 | 4.6 | 5.4 | 4.6 | **2.8** | **1.5** | 1.1 | 1.1 | 8.7 |
| BARLEY | 84 | 83.6 | 86.5 | 79.6 | 79.6 | **78.4** | 83.9 | 103.8 | 84.8 | 79.3 | 79.8 | **76.3** | 83.7 | 83.7 | 87 | 84 | 84 | 84.6 | 84.6 | 90.7 | 86 | 84.1 | **82.3** | **82.1** | 71.3 | 84.2 | 71.5 |
| CHILD | 23.7 | **20.3** | **19.9** | 15.4 | 16.8 | 14.9 | 19.1 | **17.6** | **15.2** | 15.1 | 15.3 | 14.2 | 23.7 | 23 | 21.8 | 16.6 | 16.7 | **14** | 19.7 | **16.9** | **16.5** | 16.8 | **15** | **15.4** | 10.3 | 15.9 | 17.6 |
| DIABETES | 601.2 | 646.4 | 652 | 580.9 | 580.9 | 599 | 601.2 | 778 | 631.3 | 581.3 | 614.1 | 589.3 | 602.2 | 627 | 670.3 | 625 | 625 | 646.5 | 602.6 | 653.7 | 648.1 | 626 | 648.6 | 642.6 | 608.2 | 733.5 | 689.1 |
| HAILFINDER | 61.5 | **52** | **45** | 51.8 | 52.2 | **34.8** | 55.1 | 55.2 | **34.3** | 46.6 | 51.2 | **30.5** | 62.9 | **53.7** | **51** | 52.1 | 52.1 | **38.6** | 56 | **42.8** | **37.6** | 52.3 | **48.4** | **37.1** | 45.3 | 36.6 | 33.3 |
| LINK | 1049.8 | 1099.6 | 1241.1 | 1019.9 | 1020.2 | 1097.5 | 1017.6 | 1159.7 | 1068.8 | 1076.8 | 1427.8 | 1363.5 | 1054.7 | 1091.9 | 1205.2 | 1037.9 | **1037.2** | **997.9** | 1032.4 | 1050.4 | 1033.5 | 1098.7 | 1141.2 | 1190 | 835.8 | 1289.2 | 1347.7 |
| MILDEW | 45.8 | **44.7** | **44.7** | 44.6 | 44.6 | **41.6** | 45.8 | 45.4 | **44.7** | 43.8 | 44.4 | **35.2** | 46 | 46.1 | 46.5 | 46 | 46 | 46 | 46 | 46.1 | 46.5 | 46 | 46 | 46 | 36.2 | 46.8 | 49.1 |
| HEPAR2 | 112.4 | **108.8** | **108.7** | 113 | 113.5 | **100.7** | 113.8 | **110.6** | **102.4** | 116.8 | **113** | **103.1** | 110.6 | **106.4** | **101.7** | 111.4 | 111.4 | **97** | 112.5 | **99.2** | **96.8** | 114.3 | **98.7** | **97.2** | 94.5 | 94 | 116.7 |
| INSURANCE | 46.8 | 44.9 | 47.3 | 47.1 | 47.1 | **43.3** | 46 | **42.9** | **41.6** | 44.9 | 43.8 | **39.5** | 46.4 | **42.9** | **42.1** | 45.5 | 45.5 | 44.6 | 45.2 | **43.2** | **40** | 43.6 | 46.5 | 43.7 | 31.2 | 49.6 | 51.8 |
| MUNIN1 | 274.8 | 288.9 | 343 | 278.7 | 278.7 | 281.2 | 275.9 | 319.1 | 298.2 | 277.9 | 279.8 | 266.6 | 275.1 | 288.5 | 340.5 | 277.6 | 277.6 | 283.1 | 275.5 | 313.8 | 301 | 278.1 | 280.5 | 278.3 | 213.5 | 277.8 | 408.1 |
| PATHFINDER | 190.3 | **188** | 199 | 194.2 | 194.2 | 236.1 | 190.9 | 198.3 | 199 | 195.4 | 216.7 | 235.7 | 190.9 | **189.2** | 190.2 | 193.3 | 193.3 | 221.4 | 190.9 | **182.8** | **188.5** | 193.8 | 210.5 | 222.5 | 258.4 | 257.4 | 337.0 |
| PIGS | 584.5 | **575.5** | 616.5 | 291.6 | 291.6 | **184.8** | 478.7 | **209** | **128.1** | 289.8 | **216.4** | **125.7** | 588 | 594.5 | 641.5 | 294.4 | 294.4 | **110.6** | 487.1 | **155.8** | **94.1** | 294.1 | **64** | **61.4** | 0.2 | 125 | 145.4 |
| WATER | 60.4 | 60.4 | 60.3 | 60.5 | 60.5 | 60.3 | 60.2 | 60.4 | 60.2 | 60.4 | 60.9 | 60.8 | 60 | 60 | 60.1 | 60 | 60 | 59.8 | 60 | 60.1 | 60 | 60 | 60.3 | 60.2 | 57.8 | 61.5 | 89.4 |
| WIN95PTS | 91.8 | **83.6** | **79.4** | 93.4 | **92** | **76.7** | 95.1 | **79.6** | **70.4** | 88.8 | 93 | **73.2** | 89 | **80.4** | **68.6** | 90.9 | **89.5** | **76** | 93.6 | **74.3** | **68** | 88.1 | **81.9** | **69.3** | 56.4 | 60.8 | 192.2 |

TABLE 3.11: The average SHD of PDAG for GS, IAMB, MMPC, and SI-HITON-PC and DAG for the corresponding hybrid algorithms, using the AND-rule("A"), **symC**("C") and **symG**("G") with 5,000 samples.

| | PDAG | | | | | | | | | | | | DAG | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GS | | | IAMB | | | MMPC | | | SI-HITON-PC | | | GS | | | IAMB | | | MMPC | | | SI-HITON-PC | | | FGES | HC | FastCHC |
| | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | | | |
| ALARM | 41.2 | **37.7** | 40.2 | 30.2 | 30.4 | **21.5** | 33.3 | **30.6** | **23.7** | 30.2 | **24.9** | **20** | 40.5 | **36.1** | 38 | 25.9 | 25.2 | **17.6** | 32.3 | **26.8** | **20.9** | 26.2 | **19.3** | **16** | 5.8 | 23 | 24.5 |
| ANDES | 242 | **124.7** | **122.6** | 132.7 | 132.6 | **90.9** | 136.2 | 151.9 | **95.5** | 121.4 | 189.4 | **109.5** | 236.2 | **113.1** | **110.6** | 121.3 | 121.2 | **81.4** | 123.2 | **101.5** | **80.2** | 101.4 | **98** | **79.9** | 68.8 | 79.5 | 73.7 |
| ASIA | 6.3 | 6.3 | 8.3 | 6.4 | 6.4 | **1.9** | 4.9 | 4.6 | 6 | 5 | **2.2** | **2.3** | 5 | 5 | 5.8 | 5.1 | 5.1 | **0.8** | 3.9 | 4 | 5.8 | 3.9 | **1.4** | **0.8** | 0.7 | 0.8 | 5.0 |
| BARLEY | 78.9 | 78.6 | 79.1 | 74.3 | 74.3 | 75.2 | 80 | 88.3 | **76.9** | 74.3 | 82.7 | **71.2** | 80.6 | **79.8** | 79.7 | 72 | 72 | **67.8** | 81.2 | **79** | **78.9** | 75.1 | **71.7** | **64.5** | 51.8 | 62.6 | 63.3 |
| CHILD | 23.8 | **19.4** | **15.3** | 15.7 | 15.7 | **11.1** | 12.8 | 11.7 | **6.6** | 10.7 | 11.8 | 10.2 | 23.9 | **18.6** | **17.7** | 16.6 | 16.6 | **9** | 18.1 | **13.9** | **13.2** | 16 | **13.2** | **13.2** | 0.5 | 12.9 | 15.0 |
| DIABETES | 599.3 | 660.8 | 747.6 | 597.5 | **593.5** | 638.4 | 601.5 | 720.4 | 634.5 | 589.2 | 800.3 | 643.9 | 599.3 | 661.1 | 745.3 | 599.5 | **596.5** | 635.9 | 601.5 | 692.9 | 636.6 | 597.6 | 721.1 | 665.1 | 376 | 592.9 | 577.5 |
| HAILFINDER | 59.9 | **51.1** | **45.1** | 42.5 | 42.6 | **27.2** | 48.6 | 47.8 | **31.4** | 42.3 | 49.1 | **21.8** | 58.5 | **50** | **44.9** | 41.1 | 41.1 | **22.4** | 47.3 | **41.1** | **26.3** | 41.1 | **37.7** | **22.3** | 36.8 | 35.2 | 30.7 |
| LINK | 1002.3 | 1043.6 | 1232.5 | 872.1 | 876.1 | 922.5 | 895.4 | 1159.3 | 969.1 | 842.5 | 1157.5 | 1053.5 | 1008.7 | 1047.6 | 1193.4 | 932.4 | **930.2** | **814.2** | 960.5 | 986.3 | **933.9** | 888.2 | **829.4** | 880.1 | 756.8 | 1302.9 | 1097.5 |
| MILDEW | 44.8 | **41.9** | **42.4** | 44.3 | 40.3 | **38** | 44.9 | **42.7** | **41.2** | 44.2 | 49.9 | **39.7** | 46 | **44.9** | **43.9** | 45.8 | 45.8 | 43.8 | 46 | **45.1** | **44.5** | 45.6 | 49.5 | 47 | 25.9 | 46 | 49.1 |
| HEPAR2 | 103.5 | **95.9** | **91.6** | 98.1 | 98.2 | **85** | 99.4 | **89.6** | **82.9** | 102.2 | **94.5** | **82.7** | 104.4 | **94.5** | **71.3** | 97.5 | 97.5 | **66.8** | 98.4 | **69.3** | **68.3** | 98 | **67.1** | **67.8** | 68.7 | 67.4 | 63.9 |
| INSURANCE | 48.5 | **43.4** | **45.9** | 45.2 | 46.3 | **29.8** | 47.2 | **39.3** | **30.1** | 40.7 | 43.6 | **35.2** | 45.7 | 45.1 | 45.5 | 44.2 | 45.1 | **39.8** | 43.7 | 42.1 | **38.3** | 42.9 | 49.1 | 45.3 | 28.1 | 46.3 | 48.5 |
| MUNIN1 | 274.7 | 290.5 | 366.7 | 265.7 | 265.5 | 281.3 | 272.2 | 310.6 | 300.8 | 275.1 | 326.2 | 311.1 | 275.2 | 292.1 | 358 | 269.9 | 270.2 | 286.6 | 272.1 | 318.3 | 307.3 | 276.8 | 324.8 | 315.3 | 209.5 | 234.9 | 316.8 |
| PATHFINDER | 187.1 | **182.4** | 193.6 | 190.6 | 191.7 | 208.5 | 187.8 | 192.5 | 183.2 | 191.5 | 203.1 | 204.6 | 186.8 | **182.6** | **184** | 190.7 | 191.6 | 186.7 | 187.8 | **171.1** | **168.8** | 192.2 | **181.2** | 199.5 | 247.3 | 263.5 | 277.7 |
| PIGS | 588.4 | 605.8 | 693.9 | 224.3 | 224.3 | **71.9** | 326.4 | **45.8** | **18.5** | 222.1 | **141.5** | **45.8** | 590.3 | 617.6 | 684 | 234.5 | 234.5 | **78.8** | 359.1 | **51.5** | **47.3** | 253 | **65.2** | **58.6** | 0 | 129.6 | 136.5 |
| WATER | 59.1 | 59.5 | 62.6 | 60 | 60 | 59.5 | 60 | 59.4 | 60 | 60.1 | 61.2 | 61.4 | 60 | 60 | **58.6** | 58.6 | 58.6 | **56.2** | 60 | **58** | 59.1 | 59.5 | 59.1 | **57.5** | 48.9 | 58.1 | 79.5 |
| WIN95PTS | 81.2 | **72.5** | **65** | 78.4 | 76.8 | **56.6** | 76.6 | **65.5** | **55.4** | 68 | 72.7 | **51.4** | 77 | **70.6** | **61.5** | 77.5 | **75.8** | **58** | 73 | **60.6** | **53.2** | 66.9 | **63.3** | **50.5** | 38.9 | 48.2 | 99.6 |

TABLE 3.12: The average BIC scores per sample for the DAG extension of PDAG from GS, IAMB, MMPC, and SI-HITON-PC and DAG from the corresponding hybrid algorithms, using the AND-rule("A"), **symC**("C") and **symG**("G") with 500 samples. "OOM" is the short name of "Out of Memory". "n.e." is the short name of "no extension".

| | PDAG | | | | | | | | | | | | DAG | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GS | | | IAMB | | | MMPC | | | SI-HITON-PC | | | GS | | | IAMB | | | MMPC | | | SI-HITON-PC | | | FGES | HC | FastCHC |
| | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | | | |
| ALARM | -17.77 | -28.73 | -53.01 | -15.72 | -15.72 | n.e. | -16.9 | -16.68 | **-14.89** | -15.81 | **-13.97** | -13.84 | -17.56 | **-16.26** | **-15.06** | -15.57 | -15.57 | **-13.2** | -16.74 | **-14.23** | **-14.33** | -15.38 | **-13.37** | **-13.49** | -12.91 | -12.7 | -35.07 |
| ANDES | -109.05 | **-103.42** | n.e. | -104.23 | **-104.16** | n.e. | -104.49 | n.e. | n.e. | -102.6 | n.e. | n.e. | -108.88 | **-103** | **-100.04** | -103.8 | **-103.71** | **-99.42** | -103.86 | **-99.85** | **-98.08** | -101.94 | **-99.21** | **-97.51** | -98.52 | -96.94 | -170.12 |
| ASIA | -2.53 | -2.53 | -2.55 | -2.53 | -2.53 | **-2.33** | -2.53 | **-2.5** | **-2.49** | -2.46 | **-2.43** | **-2.33** | -2.52 | -2.52 | **-2.49** | -2.52 | -2.52 | **-2.34** | -2.52 | -2.51 | **-2.49** | -2.45 | **-2.42** | **-2.34** | -2.34 | -2.34 | -3.04 |
| BARLEY | -82.13 | -81.97 | **-78.65** | -83.04 | -83.04 | **-81.3** | -81.24 | -143.65 | **-78.52** | -82.46 | -82.94 | **-80.04** | -81.56 | -81.4 | **-77.97** | -75.96 | -75.96 | **-74.36** | -80.69 | **-78.42** | **-77.59** | -75.88 | **-75.7** | **-73.86** | -85.73 | -73.12 | -121.63 |
| CHILD | -15.85 | -15.61 | -27.75 | -13.92 | -13.92 | **-13.6** | -15.21 | **-14.45** | **-13.85** | -13.99 | n.e. | -13.42 | -15.83 | **-15.43** | **-14.41** | -13.81 | -13.81 | **-13.37** | -15.2 | **-14.18** | **-13.64** | -13.8 | **-13.39** | **-13.37** | -13.24 | -13.33 | -15.87 |
| DIABETES | -753.28 | **-737.26** | -760.32 | -709.31 | -709.31 | **-671.2** | -752.52 | -1731.76 | -1003.09 | -709.05 | -708.88 | **-667.15** | -751.34 | **-734.87** | **-712.4** | -698.19 | -698.19 | **-658.43** | -750.45 | **-712.84** | **-709.28** | -697.99 | **-693.37** | **-651.11** | -1070.75 | -625.36 | -2279.25 |
| HAILFINDER | -66.01 | **-62.08** | **-61.57** | -61.45 | -61.45 | **-57.35** | -63.19 | -66.96 | -63.33 | -61.47 | **-61.29** | -60.47 | -65.66 | **-61.48** | **-60.27** | -61.17 | -61.17 | **-55.94** | -63.02 | **-59.18** | **-58.15** | -61.09 | **-60.76** | **-55.83** | -55.81 | -55.72 | -231.31 |
| LINK | -404.78 | -404.46 | -564.91 | -382.3 | -382.3 | n.e. | -388.91 | n.e. | n.e. | -375.11 | n.e. | n.e. | -404.69 | **-401.51** | **-387.75** | -378.66 | -378.64 | **-351.5** | -388.17 | **-361.1** | **-351.13** | -369.24 | **-341.64** | **-340.72** | -329.05 | -310.44 | -899.40 |
| MILDEW | -66.23 | -66.19 | **-65.92** | -65.41 | -65.41 | **-65.19** | -66.18 | -66.41 | -66.01 | -65.42 | -65.64 | **-64.12** | -64.23 | **-64.16** | **-63.91** | -63.15 | -63.15 | **-62.84** | -64.17 | **-64.13** | **-63.88** | -63.15 | -63.15 | **-61.76** | -114.94 | -61.73 | -279.64 |
| HEPAR2 | -34.24 | **-34.1** | -33.77 | -34.01 | -34.01 | -33.99 | -34.15 | n.e. | n.e. | -33.98 | n.e. | -34.62 | -34.21 | **-34.01** | **-33.72** | -33.94 | -33.94 | **-33.69** | -34.08 | **-33.7** | **-33.67** | -33.9 | **-33.67** | **-33.64** | -33.7 | -33.63 | -47.05 |
| INSURANCE | -18.56 | **-18.1** | -18.06 | -17.69 | n.e. | -17.03 | -18.49 | n.e. | **-16.73** | -17.85 | n.e. | n.e. | -18.39 | **-17.92** | **-16.66** | -17.43 | -17.43 | **-16.37** | -18.47 | **-16.31** | **-16.29** | -17.13 | **-16.34** | **-15.8** | -15.64 | -15.52 | -25.59 |
| MUNIN1 | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | -99.59 | **-94.74** | **-86.48** | -91.63 | -91.63 | **-75.66** | -95.84 | **-78.41** | **-78.69** | -90.01 | **-85.91** | **-69.75** | -96.86 | -65.69 | -3077.42 |
| PATHFINDER | -62.47 | -62.33 | **-61.04** | -56.1 | -56.1 | -45.31 | -61.97 | -62.07 | **-60.78** | -54.5 | **-52.38** | n.e. | -61.84 | -61.69 | **-59.81** | -55.78 | -55.78 | **-44.1** | -61.79 | **-60.3** | **-59.58** | -54.18 | **-51.84** | **-42.19** | -46.2 | -38.41 | -428.13 |
| PIGS | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | -461.26 | **-459.98** | **-455.65** | -404.31 | **-404.25** | **-371.86** | -426.62 | **-377.02** | **-375.49** | -412.36 | **-369.19** | **-366.61** | -363.23 | -365.91 | -451.57 |
| WATER | -14.72 | **-14.47** | **-14.35** | -14.47 | -14.47 | **-14.3** | -14.71 | **-14.44** | **-14.31** | -14.46 | -14.47 | **-14.3** | -14.69 | **-14.44** | **-14.33** | -14.45 | -14.45 | **-14.28** | -14.69 | **-14.41** | **-14.29** | -14.45 | -14.45 | **-14.28** | -14.4 | -14.28 | -242.89 |
| WIN95PTS | -15.29 | **-14.49** | **-13.94** | -14.49 | **-14.48** | -13.68 | -14.56 | **-12.86** | n.e. | -13.95 | -13.33 | n.e. | -15.19 | **-14.37** | **-13.5** | -14.38 | **-14.38** | **-12.98** | -14.46 | **-12.68** | **-11.91** | -13.67 | **-13.09** | **-11.59** | -11.14 | -10.86 | -215.22 |

TABLE 3.13: The average BIC scores per sample for the DAG extension of PDAG from GS, IAMB, MMPC, and SI-HITON-PC and DAG from the corresponding hybrid algorithms, using the AND-rule("A"), **symC**("C") and **symG**("G") with 1,000 samples. "OOM" is the short name of "Out of Memory". "n.e." is the short name of "no extension".

| | PDAG | | | | | | | | | | | | DAG | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GS | | | IAMB | | | MMPC | | | SI-HITON-PC | | | GS | | | IAMB | | | MMPC | | | SI-HITON-PC | | | FGES | HC | FastCHC |
| | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | | | |
| ALARM | -16.89 | -63.96 | -87.86 | -14.63 | -14.63 | n.e. | -15.76 | -14.37 | -13.57 | -14.37 | n.e. | n.e. | -16.71 | **-14.72** | **-13.72** | -14.16 | -14.15 | **-12.03** | -15.53 | **-13.23** | **-13.09** | -13.99 | **-12.34** | **-12.03** | -11.96 | -11.86 | -12.95 |
| ANDES | -107.01 | **-100.7** | **-98.7** | -101.81 | **-101.78** | n.e. | -101.74 | n.e. | n.e. | -100.91 | n.e. | n.e. | -106.89 | **-100.46** | **-97.76** | -101.52 | **-101.49** | **-97.86** | -101.27 | **-97.89** | **-96.33** | -100.34 | **-97.72** | **-95.69** | -96.16 | -95.3 | -96.70 |
| ASIA | -2.5 | -2.5 | -2.5 | -2.5 | -2.5 | **-2.32** | -2.5 | **-2.48** | **-2.44** | -2.5 | **-2.45** | **-2.3** | -2.49 | -2.49 | **-2.44** | -2.49 | -2.49 | **-2.3** | -2.48 | -2.48 | **-2.44** | -2.48 | **-2.43** | **-2.3** | -2.3 | -2.29 | -2.35 |
| BARLEY | -80.77 | -81 | -81 | -77.02 | -77.02 | **-74.45** | -79.47 | n.e. | -80.62 | -74.91 | -75.39 | **-72.73** | -80.66 | -80.63 | **-77.16** | -73.77 | -73.77 | **-70.83** | -79.32 | **-77.54** | **-76.58** | -72.37 | **-71.81** | **-69.84** | -77.99 | -69.06 | -79.51 |
| CHILD | -15.59 | **-15.04** | -59.33 | -13.2 | -13.2 | **-12.93** | -14.03 | n.e. | **-13.04** | -13.22 | n.e. | **-12.97** | -15.58 | **-14.84** | **-14.1** | -13.17 | -13.16 | **-12.99** | -14.03 | **-13.27** | **-13.06** | -13.18 | **-12.97** | **-12.97** | -12.83 | -12.92 | -12.99 |
| DIABETES | -746.38 | -53738.18 | -1983.62 | -721.68 | **-720.88** | **-690.14** | -721.26 | n.e. | n.e. | -721.01 | **-706.03** | **-686.97** | -745.87 | **-688.58** | **-660.74** | -627.81 | **-627.01** | **-596.09** | -720.72 | **-675.33** | **-671.42** | -627.25 | **-611.81** | **-592.65** | -809.66 | -555.45 | -1380.91 |
| HAILFINDER | -64.76 | **-61.05** | **-61.34** | -59.43 | -59.44 | **-55.69** | -61.67 | -61.21 | **-55.61** | -59.47 | -59.36 | **-55.4** | -64.73 | **-60.72** | **-59.34** | -59.3 | -59.3 | **-53.58** | -61.54 | **-55.6** | **-54.95** | -59.23 | **-58.75** | **-53.47** | -53.33 | -53.27 | -67.87 |
| LINK | -399.9 | -469.74 | n.e. | -376.85 | -376.68 | n.e. | -379.54 | n.e. | n.e. | -366.89 | n.e. | n.e. | -399.87 | **-392.33** | **-376.65** | -373.1 | **-372.93** | **-343.13** | -379.48 | **-330.9** | **-327.78** | -358.38 | **-329.52** | **-328.16** | -308.42 | -288.9 | -348.05 |
| MILDEW | -62.09 | **-61.37** | **-60.92** | -60.99 | -60.99 | -63.18 | -62.04 | **-61.17** | **-60.92** | -61.04 | -61.13 | -61.03 | -61.7 | **-60.6** | **-60.17** | -60.18 | -60.18 | -61.3 | -61.65 | **-60.44** | **-60.17** | -60.17 | **-59.94** | **-58.95** | -88.79 | -58.94 | -87.79 |
| HEPAR2 | -33.9 | **-33.75** | -34.11 | -33.63 | -33.63 | -34 | -33.78 | n.e. | n.e. | -33.61 | n.e. | -33.07 | -33.83 | **-33.61** | **-33.37** | -33.57 | -33.57 | **-33.31** | -33.7 | **-33.33** | **-33.28** | -33.55 | **-33.31** | **-33.28** | -33.29 | -33.26 | -33.56 |
| INSURANCE | -18.15 | **-17.01** | -16.22 | -17.34 | -17.29 | n.e. | -17.75 | n.e. | n.e. | -16.43 | n.e. | **-15.08** | -17.62 | **-16.55** | **-15.76** | -16.6 | -16.6 | **-15.27** | -17.49 | **-15.46** | **-15.42** | -15.81 | **-14.97** | **-14.91** | -14.68 | -14.77 | -15.28 |
| MUNIN1 | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | -97.65 | **-91.26** | **-81.04** | -87.62 | -87.58 | **-71.54** | -92.07 | **-69.8** | **-69.58** | -83.75 | **-73.6** | **-63.46** | -74.56 | -58 | -143.65 |
| PATHFINDER | -59.77 | **-59.4** | -89.82 | -55.29 | -55.29 | -44.67 | -59.62 | n.e. | -305.87 | -55.79 | **-48.73** | n.e. | -59.71 | **-59.29** | **-51.19** | -55.23 | -55.23 | **-42.57** | -59.62 | **-49.45** | **-50.82** | -55.09 | **-48.03** | **-40.42** | -42.17 | -35.25 | -56.75 |
| PIGS | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | -455.84 | **-439.11** | **-430.35** | -382.6 | -382.6 | **-357.68** | -422.07 | **-363.44** | **-355.49** | -382.23 | **-351.59** | **-351.24** | -348.44 | -349.52 | -350.27 |
| WATER | -14.15 | -14.13 | **-14.08** | -13.82 | -13.82 | **-13.75** | -14.03 | **-13.82** | **-13.78** | -13.82 | **-13.82** | **-13.75** | -14.14 | -14.12 | **-14.07** | -13.81 | -13.81 | **-13.75** | -14.02 | **-13.82** | **-13.77** | -13.81 | **-13.8** | **-13.74** | -13.83 | -13.73 | -22.73 |
| WIN95PTS | -14.6 | **-13.59** | **-12.91** | -13.66 | -13.57 | -12.6 | -13.81 | **-12.29** | -11.55 | -13.26 | -12.19 | -11.09 | -14.37 | **-13.5** | **-12.51** | -13.52 | **-13.38** | **-12.53** | -13.47 | **-12.21** | **-11.2** | -12.96 | **-12.27** | **-10.94** | -10.44 | -10.24 | -13.43 |

TABLE 3.14: The average BIC scores per sample for the DAG extension of PDAG from GS, IAMB, MMPC, and SI-HITON-PC and DAG from the corresponding hybrid algorithms, using the AND-rule("A"), **symC**("C") and **symG**("G") with 5,000 samples. "OOM" is the short name of "Out of Memory". "n.e." is the short name of "no extension".

| | PDAG | | | | | | | | | | | | DAG | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GS | | | IAMB | | | MMPC | | | SI-HITON-PC | | | GS | | | IAMB | | | MMPC | | | SI-HITON-PC | | | FGES | HC | FastCHC |
| | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | | | |
| ALARM | -16.07 | -24.47 | -84.62 | -13.29 | -13.35 | n.e. | -13.85 | **-12.79** | **-12.6** | n.e. | -11.53 | n.e. | -15.94 | **-13.25** | **-12.77** | -12.86 | **-12.84** | **-11.49** | -13.75 | **-12.15** | **-12.13** | -14.1 | **-11.77** | **-11.51** | -10.85 | -10.94 | -11.17 |
| ANDES | -104.57 | **-97.54** | **-95.72** | -98.62 | -98.62 | n.e. | -98.69 | n.e. | n.e. | -97.7 | n.e. | n.e. | -104.37 | **-97.23** | **-95.53** | -98.41 | -98.41 | **-95.66** | -98.18 | **-95.85** | **-94.06** | -97.22 | **-96.04** | **-93.68** | -93.81 | -93.63 | -93.82 |
| ASIA | -2.49 | -2.49 | **-2.45** | -2.49 | -2.49 | **-2.32** | -2.47 | **-2.43** | **-2.4** | -2.47 | **-2.36** | **-2.33** | -2.46 | -2.46 | **-2.4** | -2.46 | -2.46 | **-2.26** | -2.44 | -2.43 | **-2.4** | -2.44 | **-2.35** | **-2.26** | -2.26 | -2.26 | -2.27 |
| BARLEY | -77.65 | **-76.62** | -117.65 | -72.44 | **-70.96** | -82.74 | -72.26 | n.e. | -134.96 | -71.08 | n.e. | n.e. | -77.57 | **-76.39** | **-69.85** | -71.47 | **-69.93** | **-64.18** | -71.97 | **-65.28** | **-65.38** | -70.82 | **-65.89** | **-63.5** | -64.96 | -59.4 | -115.11 |
| CHILD | -15.3 | **-13.94** | **-13.44** | -13.4 | -13.4 | -12.54 | -13.63 | n.e. | **-12.89** | -13.44 | n.e. | **-12.58** | -15.23 | **-13.8** | **-13.06** | -13.3 | -13.3 | **-12.58** | -13.44 | **-12.53** | **-12.52** | -13.36 | **-12.53** | **-12.52** | -12.38 | -12.42 | -12.47 |
| DIABETES | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | -721.73 | **-660.66** | **-612.15** | -685.53 | **-647.38** | **-539.45** | -701.46 | **-551.96** | **-551.92** | -683.17 | **-567.57** | **-566.59** | -557.63 | -402.21 | -626.58 |
| HAILFINDER | -63.21 | **-58.18** | **-56.97** | -57.68 | -57.68 | -50.56 | -59.1 | **-55.29** | **-53.08** | -57.69 | -57.32 | **-50.79** | -63.16 | **-57.66** | **-56.15** | -57.66 | -57.66 | **-50.59** | -59.04 | **-51.52** | **-50.92** | -57.64 | **-56.8** | **-50.54** | -51.26 | -50.43 | -54.02 |
| LINK | -392.87 | -912.39 | n.e. | -366.52 | -366.95 | n.e. | -367.47 | n.e. | n.e. | n.e. | n.e. | n.e. | -393.44 | **-373.42** | **-352.76** | -366.58 | **-365.92** | **-303.22** | -369.45 | **-311.97** | **-306.31** | -345.56 | **-285.65** | **-285.14** | -285.41 | -257.93 | -309.95 |
| MILDEW | -59.02 | **-56.39** | **-55.67** | -58.61 | **-57.58** | **-58.24** | -59.66 | -153.27 | -143.25 | -59.11 | -60.49 | **-56.69** | -59.01 | **-56.01** | **-55.57** | -58.54 | **-56.95** | **-55.88** | -59.66 | **-55.75** | **-55.54** | -59.1 | **-55.43** | **-54.8** | -81.28 | -52.62 | -67.44 |
| HEPAR2 | -33.38 | **-33.17** | n.e. | -33.15 | -33.15 | n.e. | -33.12 | n.e. | n.e. | -33.11 | n.e. | n.e. | -33.35 | **-33.08** | **-32.88** | -33.07 | -33.07 | **-32.89** | -33.13 | **-32.84** | **-32.79** | -33.05 | **-32.8** | **-32.79** | -32.79 | -32.78 | -32.82 |
| INSURANCE | -17.95 | **-16.03** | n.e. | -16 | -16.07 | n.e. | -16.22 | n.e. | n.e. | -15.8 | n.e. | n.e. | -17.28 | **-15.34** | **-15.07** | -15.11 | -15.08 | **-13.94** | -15.6 | **-14.1** | **-14.19** | -15.42 | **-13.99** | **-13.9** | -13.62 | -13.61 | -13.80 |
| MUNIN1 | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | -95.28 | **-87.55** | **-74.62** | -84.78 | **-81.17** | **-63.47** | -88.42 | **-61.69** | **-62.99** | -90.82 | **-68.74** | **-69.32** | -52.97 | -46.73 | -58.60 |
| PATHFINDER | -56.69 | **-54.34** | -16764.74 | -56.69 | **-56.68** | -89.16 | -56.75 | n.e. | **-50.19** | -56.85 | n.e. | n.e. | -56.56 | **-54.16** | **-44** | -56.63 | **-56.62** | **-45.91** | -56.75 | **-40.31** | **-47.02** | -56.81 | **-40.3** | **-41.05** | -35.5 | -30.31 | -40.40 |
| PIGS | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | -453.38 | **-442.84** | **-432.16** | -364.1 | -364.1 | **-341.2** | -386.97 | **-337.73** | **-336.94** | -367.33 | **-339.08** | **-338.4** | -334.82 | -335.04 | -335.43 |
| WATER | -13.57 | **-13.4** | -13.46 | n.e. | n.e. | n.e. | -13.43 | **-13.24** | **-13.19** | -13.21 | -13.15 | -13.17 | -13.54 | **-13.38** | **-13.35** | -13.18 | -13.18 | **-13.17** | -13.43 | **-13.22** | **-13.18** | -13.19 | **-13.15** | **-13.16** | -13.18 | -13.13 | -14.62 |
| WIN95PTS | -13.84 | **-13.11** | -12.15 | -12.5 | **-12.36** | **-11.53** | -12.44 | **-11.43** | -10.78 | -12.2 | -11.28 | **-10.7** | -13.52 | **-12.7** | **-11.2** | -12.23 | **-12.14** | **-11.41** | -12.11 | **-11.18** | **-10.25** | -11.86 | **-11.19** | **-10.12** | -9.61 | -9.45 | -9.75 |

TABLE 3.15: The average BDeu scores per sample for GS, IAMB, MMPC, and SI-HITON-PC using the AND-rule("A"), **symC**("C") and **symG**("G") with 500 samples. "OOM" is the short name of "Out of Memory". "n.e." is the short name of "no extension".

| | PDAG | | | | | | | | | | | | DAG | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GS | | | IAMB | | | MMPC | | | SI-HITON-PC | | | GS | | | IAMB | | | MMPC | | | SI-HITON-PC | | | FGES | HC | FastCHC |
| | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | | | |
| ALARM | -17.6 | **-17.03** | **-16.35** | -15.33 | -15.33 | n.e. | -16.78 | **-14.45** | **-14.38** | -15.47 | **-12.93** | -13.46 | -17.49 | **-16.13** | **-14.84** | -15.26 | -15.26 | **-12.65** | -16.64 | **-13.97** | **-14.01** | -15.14 | **-12.81** | **-13.07** | -11.71 | -12.07 | -13.86 |
| ANDES | -108.97 | **-103.22** | n.e. | -104.08 | **-104** | n.e. | -104.31 | n.e. | n.e. | -102.42 | n.e. | n.e. | -108.79 | **-102.8** | **-99.79** | -103.62 | **-103.53** | **-99.16** | -103.67 | **-99.59** | **-97.75** | -101.75 | **-98.95** | **-97.19** | -98.24 | -96.6 | -103.20 |
| ASIA | -2.52 | -2.52 | -2.53 | -2.52 | -2.52 | **-2.31** | -2.52 | **-2.49** | **-2.48** | -2.45 | **-2.42** | **-2.31** | -2.51 | -2.51 | **-2.48** | -2.51 | -2.51 | **-2.32** | -2.5 | -2.49 | **-2.48** | -2.44 | **-2.41** | **-2.32** | -2.32 | -2.32 | -2.50 |
| BARLEY | -82.33 | -82.16 | **-77.26** | -73.49 | -73.49 | **-70.35** | -81.24 | -82.45 | **-76.85** | -72.67 | -73.18 | **-69.91** | -81.57 | -81.39 | **-76.53** | -74.49 | -74.49 | **-71.99** | -80.5 | **-76.95** | **-76.06** | -74.41 | **-74.24** | **-71.44** | -66.87 | -70.29 | -80.74 |
| CHILD | -15.88 | **-15.58** | **-15.19** | -13.85 | -13.85 | **-13.51** | -15.23 | **-14.47** | **-13.76** | -13.92 | n.e. | -13.45 | -15.87 | **-15.47** | **-14.48** | -13.83 | -13.83 | **-13.39** | -15.22 | **-14.2** | **-13.66** | -13.82 | **-13.4** | **-13.37** | -13.24 | -13.33 | -15.07 |
| DIABETES | -750.06 | **-730.91** | **-690.88** | -627.77 | -627.77 | **-571.12** | -748.39 | **-705.63** | **-682.19** | -627.07 | **-596.42** | **-558.12** | -750.4 | **-730.64** | **-689.16** | -672.34 | -672.34 | **-617.95** | -748.85 | **-690.2** | **-685.62** | -672.05 | **-644.15** | **-608.88** | -469.45 | -570.51 | -644.46 |
| HAILFINDER | -65.87 | **-61.22** | **-60.3** | -60.66 | -60.66 | **-55.7** | -62.97 | **-61.2** | **-59.14** | -60.69 | -60.62 | **-55.92** | -65.67 | **-60.85** | **-59.62** | -60.44 | -60.44 | **-54.91** | -62.83 | **-58.78** | **-57.7** | -60.37 | **-60.03** | **-54.79** | -54.51 | -54.58 | -62.60 |
| LINK | -404.82 | **-402.35** | -394.42 | -379.79 | -379.81 | n.e. | -388.87 | n.e. | n.e. | -372.04 | n.e. | n.e. | -405.01 | **-401.66** | **-386.91** | -378.57 | -378.54 | **-344.63** | -388.9 | **-359.72** | **-346.97** | -369.15 | **-334.44** | **-333.64** | -304.97 | -288.81 | -376.49 |
| MILDEW | -63.07 | **-62.99** | **-62.66** | -61.84 | -61.84 | **-60.95** | -63.02 | -63.02 | **-62.66** | -61.85 | -61.86 | **-59.68** | -64.32 | **-64.24** | **-63.9** | -63.07 | -63.07 | **-62.17** | -64.26 | **-64.22** | **-63.87** | -63.07 | -63.07 | **-60.89** | -55.98 | -60.81 | -67.02 |
| HEPAR2 | -34.25 | **-34.08** | -33.65 | -33.98 | -33.99 | **-33.82** | -34.14 | n.e. | n.e. | -33.96 | n.e. | -34.4 | -34.22 | **-34.01** | **-33.71** | -33.94 | -33.94 | **-33.68** | -34.08 | **-33.68** | **-33.66** | -33.9 | **-33.66** | **-33.62** | -33.69 | -33.6 | -35.35 |
| INSURANCE | -18.39 | **-17.9** | **-16.87** | -17.46 | n.e. | -16.47 | -18.37 | n.e. | **-16.42** | -17.51 | n.e. | n.e. | -18.27 | **-17.78** | **-16.35** | -17.25 | -17.25 | **-16.03** | -18.39 | **-15.92** | **-15.95** | -16.92 | **-15.97** | **-15.35** | -14.66 | -15.06 | -17.04 |
| MUNIN1 | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | -98.59 | **-93** | **-82.96** | -89.02 | -89.02 | **-68.89** | -94.51 | **-74.4** | **-74.59** | -87.28 | **-82.7** | **-62.31** | -49.66 | -57.5 | -64.83 |
| PATHFINDER | -61.15 | **-60.96** | **-59.03** | -54.87 | -54.87 | -42.63 | -61.3 | **-60.68** | **-58.61** | -53.24 | **-50.06** | n.e. | -61.46 | **-61.26** | **-58.51** | -54.87 | -54.87 | **-41.95** | -61.4 | **-59.76** | **-58.17** | -53.16 | **-50.45** | **-39.95** | -34.32 | -35.93 | -43.56 |
| PIGS | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | -461.56 | **-460.19** | **-455.83** | -400.01 | **-399.94** | **-361.84** | -424.85 | **-368.17** | **-366.32** | -408.68 | **-358.66** | **-355.61** | -350.85 | -354.09 | -433.95 |
| WATER | -14.35 | **-14.06** | **-13.88** | -14.08 | -14.08 | **-13.82** | -14.36 | **-14.02** | **-13.83** | -14.07 | **-14.07** | **-13.82** | -14.36 | **-14.07** | **-13.89** | -14.07 | -14.07 | **-13.82** | -14.38 | **-14.03** | **-13.85** | -14.07 | -14.07 | **-13.82** | -13.73 | -13.82 | -15.57 |
| WIN95PTS | -15.17 | **-14.31** | **-13.64** | -14.39 | **-14.38** | **-13.42** | -14.46 | **-12.62** | n.e. | -13.83 | **-13.12** | n.e. | -15.08 | **-14.21** | **-13.26** | -14.29 | **-14.28** | **-12.77** | -14.37 | **-12.47** | **-11.63** | -13.56 | **-12.92** | **-11.29** | -10.65 | -10.33 | -11.09 |

TABLE 3.16: The average BDeu scores per sample for GS, IAMB, MMPC, and SI-HITON-PC using the AND-rule("A"), **symC**("C") and **symG**("G") with 1,000 samples. "OOM" is the short name of "Out of Memory". "n.e." is the short name of "no extension".

| | PDAG | | | | | | | | | | | | DAG | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GS | | | IAMB | | | MMPC | | | SI-HITON-PC | | | GS | | | IAMB | | | MMPC | | | SI-HITON-PC | | | FGES | HC | FastCHC |
| | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | | | |
| ALARM | -16.78 | **-15.99** | **-15.77** | -14.37 | -14.36 | n.e. | -15.62 | -13.98 | -13.39 | -14.19 | n.e. | n.e. | -16.67 | **-14.58** | **-13.51** | -13.94 | -13.93 | **-11.68** | -15.41 | **-13** | **-12.82** | -13.77 | **-12.02** | **-11.69** | -11.18 | -11.41 | -11.41 |
| ANDES | -106.94 | **-100.58** | **-98.55** | -101.69 | **-101.66** | n.e. | -101.61 | n.e. | n.e. | -100.78 | n.e. | n.e. | -106.81 | **-100.32** | **-97.61** | -101.4 | **-101.36** | **-97.7** | -101.14 | **-97.73** | **-96.13** | -100.21 | **-97.57** | **-95.49** | -95.97 | -95.1 | -95.72 |
| ASIA | -2.5 | -2.5 | -2.49 | -2.5 | -2.5 | **-2.31** | -2.49 | **-2.48** | **-2.43** | -2.49 | **-2.44** | **-2.29** | -2.48 | -2.48 | **-2.43** | -2.48 | -2.48 | **-2.29** | -2.48 | -2.48 | **-2.43** | -2.48 | **-2.42** | **-2.29** | -2.29 | -2.28 | -2.29 |
| BARLEY | -80.79 | -80.57 | **-77.7** | -69.99 | -69.99 | **-66.09** | -79.38 | n.e. | **-76.85** | -69.18 | **-68.17** | **-65.06** | -80.62 | -80.58 | **-76.64** | -72.84 | -72.84 | **-69.55** | -79.19 | **-77.06** | **-76.22** | -71.31 | **-70.19** | **-67.49** | -62.22 | -66.52 | -63.13 |
| CHILD | -15.61 | **-15.05** | -15.18 | -13.21 | -13.21 | **-12.96** | -14.05 | n.e. | **-13.06** | -13.22 | n.e. | **-12.99** | -15.6 | **-14.87** | **-14.15** | -13.18 | -13.18 | **-13.01** | -14.05 | **-13.29** | **-13.08** | -13.19 | **-12.99** | **-12.99** | -12.86 | -12.94 | -12.99 |
| DIABETES | -744.3 | **-697.31** | **-662.29** | -545.49 | **-544.42** | **-504.35** | -717.01 | n.e. | n.e. | -544.78 | **-517.85** | **-499.48** | -744.95 | **-680.87** | **-648.36** | -579.33 | **-578.25** | **-541.22** | -717.92 | **-667.07** | **-662.4** | -578.62 | **-556.21** | **-538.01** | -399.87 | -489.36 | -459.66 |
| HAILFINDER | -64.77 | **-60.84** | **-59.89** | -59.04 | -59.04 | **-53.59** | -61.47 | **-56.48** | **-55.17** | -59.11 | -59.09 | **-53.65** | -64.72 | **-60.56** | **-59.18** | -58.93 | -58.93 | **-52.99** | -61.33 | **-55.24** | **-54.62** | -58.86 | **-58.43** | **-52.88** | -52.66 | -52.68 | -52.95 |
| LINK | -399.78 | **-394.21** | n.e. | -374.14 | -374 | n.e. | -378.63 | n.e. | n.e. | -360.48 | n.e. | n.e. | -399.92 | **-392.09** | **-375.6** | -372.61 | **-372.42** | **-338.41** | -379.52 | **-326.23** | **-322.83** | -357.47 | **-324.5** | **-322.76** | -292.47 | -269.33 | -301.86 |
| MILDEW | -60.31 | **-59.15** | **-58.53** | -58.67 | -58.67 | -58.92 | -60.26 | **-58.89** | **-58.53** | -58.61 | **-58.22** | **-56.35** | -61.7 | **-60.5** | **-59.88** | -60.03 | -60.03 | -60.38 | -61.66 | **-60.26** | **-59.88** | -60.04 | **-59.59** | **-57.83** | -52.7 | -57.83 | -54.66 |
| HEPAR2 | -33.89 | **-33.7** | -33.59 | -33.61 | -33.61 | -33.56 | -33.78 | n.e. | n.e. | -33.59 | n.e. | -33.06 | -33.83 | **-33.61** | **-33.36** | -33.57 | -33.57 | **-33.3** | -33.7 | **-33.32** | **-33.27** | -33.54 | **-33.3** | **-33.26** | -33.28 | -33.25 | -33.32 |
| INSURANCE | -18 | **-16.83** | -15.73 | -17.01 | -16.99 | n.e. | -17.64 | n.e. | n.e. | -16.21 | n.e. | **-14.77** | -17.54 | **-16.41** | **-15.53** | -16.49 | -16.49 | **-14.98** | -17.41 | **-15.21** | **-15.18** | -15.66 | **-14.62** | **-14.61** | -14.09 | -14.37 | -14.19 |
| MUNIN1 | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | -97.07 | **-90.01** | **-78.71** | -85.24 | -85.19 | **-65.67** | -91.06 | **-66.63** | **-66.72** | -81.06 | **-69.98** | **-57.26** | -45.75 | -51.63 | -47.53 |
| PATHFINDER | -59.3 | **-58.12** | **-55.15** | -54.75 | -54.75 | **-42.2** | -59.43 | n.e. | **-55.45** | -54.88 | **-46.27** | n.e. | -59.52 | **-58.43** | **-50.45** | -54.76 | -54.76 | **-41.27** | -59.43 | **-48.83** | **-49.97** | -54.36 | **-45.99** | **-38.52** | -31.88 | -33.29 | -33.47 |
| PIGS | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | -455.93 | **-438.77** | **-429.93** | -378.93 | -378.93 | **-351.62** | -420.69 | **-357.82** | **-349.22** | -378.52 | **-344.95** | **-344.57** | -341.14 | -342.15 | -342.81 |
| WATER | -13.94 | -13.9 | **-13.83** | -13.57 | -13.57 | **-13.46** | -13.82 | **-13.57** | **-13.51** | -13.57 | **-13.57** | **-13.46** | -13.93 | -13.9 | **-13.83** | -13.56 | -13.56 | **-13.47** | -13.82 | **-13.57** | **-13.51** | -13.56 | **-13.55** | **-13.46** | -13.38 | -13.44 | -13.40 |
| WIN95PTS | -14.51 | **-13.48** | **-12.72** | -13.57 | -13.48 | -12.47 | -13.74 | **-12.14** | -11.32 | -13.17 | -12.02 | -10.88 | -14.28 | **-13.39** | **-12.33** | -13.44 | **-13.3** | **-12.37** | -13.4 | **-12.07** | **-11** | -12.87 | **-12.13** | **-10.73** | -10 | -9.84 | -9.82 |

TABLE 3.17: The average BDeu scores per sample for GS, IAMB, MMPC, and SI-HITON-PC using the AND-rule("A"), **symC**("C") and **symG**("G") with 5,000 samples. "OOM" is the short name of "Out of Memory". "n.e." is the short name of "no extension".

| | PDAG | | | | | | | | | | | | DAG | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GS | | | IAMB | | | MMPC | | | SI-HITON-PC | | | GS | | | IAMB | | | MMPC | | | SI-HITON-PC | | | FGES | HC | FastCHC |
| | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | | | |
| ALARM | -16.05 | **-14.66** | **-14.61** | -13.23 | -13.28 | n.e. | -13.81 | **-12.69** | **-12.54** | n.e. | -11.44 | n.e. | -15.93 | **-13.19** | **-12.72** | -12.8 | **-12.79** | **-11.41** | -13.72 | **-12.09** | **-12.07** | -14.07 | **-11.7** | **-11.44** | -10.7 | -10.85 | -10.75 |
| ANDES | -104.54 | **-97.48** | **-95.68** | -98.58 | -98.58 | n.e. | -98.65 | n.e. | n.e. | -97.67 | n.e. | n.e. | -104.34 | **-97.19** | **-95.48** | -98.37 | -98.37 | **-95.62** | -98.14 | **-95.81** | **-94** | -97.18 | **-95.99** | **-93.63** | -93.76 | -93.58 | -93.75 |
| ASIA | -2.49 | -2.49 | **-2.45** | -2.48 | -2.48 | **-2.31** | -2.47 | **-2.43** | **-2.39** | -2.46 | **-2.36** | **-2.33** | -2.46 | -2.46 | **-2.4** | -2.46 | -2.46 | **-2.26** | -2.44 | -2.43 | **-2.39** | -2.44 | **-2.35** | **-2.26** | -2.26 | -2.26 | -2.26 |
| BARLEY | -77.28 | **-76.18** | **-71.46** | -69.92 | **-68.38** | **-64.15** | -71.45 | n.e. | **-65.9** | -69.04 | n.e. | n.e. | -77.2 | **-75.95** | **-68.6** | -69.69 | **-68.11** | **-61.12** | -71.23 | **-63.49** | **-63.57** | -68.84 | **-62.73** | **-59.74** | -53.26 | -55 | -53.46 |
| CHILD | -15.31 | **-13.93** | **-13.4** | -13.4 | -13.4 | -12.55 | -13.63 | n.e. | **-12.87** | -13.44 | n.e. | **-12.59** | -15.24 | **-13.8** | **-13.08** | -13.3 | -13.3 | **-12.6** | -13.44 | **-12.54** | **-12.53** | -13.36 | **-12.54** | **-12.53** | -12.39 | -12.43 | -12.48 |
| DIABETES | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | -720.5 | **-657.65** | **-606.39** | -683.36 | **-638.36** | **-497.44** | -699.51 | **-538.95** | **-537.65** | -680.28 | **-555.3** | **-548.44** | -256.86 | -271.76 | -352.23 |
| HAILFINDER | -63.2 | **-57.83** | **-56.51** | -57.59 | -57.59 | -50.41 | -59.04 | **-53.01** | **-51.91** | -57.61 | -57.28 | **-50.68** | -63.16 | **-57.58** | **-56.07** | -57.57 | -57.57 | **-50.43** | -58.97 | **-51.39** | **-50.76** | -57.55 | **-56.74** | **-50.38** | -50.27 | -50.28 | -50.37 |
| LINK | -392.75 | -382.54 | n.e. | -365.28 | -365.68 | n.e. | -366.55 | n.e. | n.e. | n.e. | n.e. | n.e. | -393.4 | **-373.16** | **-352.31** | -366.01 | **-365.33** | **-300.79** | -369.11 | **-309.89** | **-304.03** | -344.39 | **-282.2** | **-281.76** | -278.2 | -250.08 | -281.23 |
| MILDEW | -58.58 | **-55.41** | **-54.84** | -58.15 | **-56.35** | **-55.51** | -59.24 | **-57.41** | **-56.83** | -58.66 | **-55.05** | **-53.97** | -58.58 | **-55.36** | **-54.88** | -58.09 | **-56.33** | **-54.98** | -59.23 | **-55.03** | **-54.8** | -58.65 | **-54.38** | **-53.76** | -47.05 | -50.87 | -49.75 |
| HEPAR2 | -33.38 | **-33.15** | n.e. | -33.14 | -33.14 | n.e. | -33.11 | n.e. | n.e. | -33.09 | n.e. | n.e. | -33.35 | **-33.08** | **-32.87** | -33.06 | -33.06 | **-32.88** | -33.12 | **-32.83** | **-32.79** | -33.05 | **-32.8** | **-32.78** | -32.79 | -32.78 | -32.80 |
| INSURANCE | -17.87 | **-15.82** | n.e. | -15.86 | -15.87 | n.e. | -16.18 | n.e. | n.e. | -15.69 | n.e. | n.e. | -17.25 | **-15.29** | **-15.01** | -15.05 | **-15.01** | **-13.82** | -15.56 | **-14** | **-14.11** | -15.37 | **-13.88** | **-13.79** | -13.42 | -13.47 | -13.48 |
| MUNIN1 | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | -95.08 | **-87.12** | **-73.58** | -84.2 | **-80.38** | **-61.7** | -88.06 | **-60.14** | **-61.5** | -90.4 | **-67.49** | **-68.05** | -41.09 | -43.28 | -41.76 |
| PATHFINDER | -56.41 | **-53.77** | **-50.1** | -56.48 | **-56.47** | **-49.7** | -56.58 | n.e. | **-48.64** | -56.76 | n.e. | n.e. | -56.38 | **-53.68** | **-41.66** | -56.43 | **-56.42** | **-44.68** | -56.58 | **-39.71** | **-45.87** | -56.73 | **-39.74** | **-37.87** | -27.79 | -28.7 | -28.55 |
| PIGS | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | OOM | -453.38 | **-442.79** | **-432.09** | -362.87 | -362.87 | **-339.43** | -386.17 | **-335.89** | **-335.08** | -366.18 | **-337.27** | **-336.57** | -332.83 | -333.05 | -333.28 |
| WATER | -13.5 | **-13.32** | **-13.28** | n.e. | n.e. | n.e. | -13.37 | **-13.17** | **-13.11** | -13.14 | **-13.05** | -13.08 | -13.47 | **-13.31** | **-13.27** | -13.1 | -13.1 | **-13.07** | -13.37 | **-13.14** | **-13.1** | -13.11 | **-13.06** | **-13.07** | -12.97 | -13.02 | -13.00 |
| WIN95PTS | -13.81 | **-13.07** | -12.07 | -12.47 | **-12.33** | **-11.48** | -12.41 | **-11.37** | -10.69 | -12.16 | -11.2 | **-10.61** | -13.49 | **-12.67** | **-11.15** | -12.2 | **-12.1** | **-11.35** | -12.08 | **-11.13** | **-10.18** | -11.83 | **-11.13** | **-10.05** | -9.37 | -9.31 | -9.32 |

TABLE 3.18: The average SHD of PDAG using the AND-rule("A"), **symC**("C") and **symG**("G") for the PDAG learned from the SLL algorithm and the DAG learned from the hybrid algorithm (SLL + hill-climbing).

| | PDAG | | | | | | | | | DAG | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Sample size | 500 | | | 1000 | | | 5000 | | | 500 | | | 1000 | | | 5000 | | |
| | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G |
| ALARM | 29.4 | 28.2 | 29.3 | 22.4 | 21.6 | 23.5 | 10.6 | 12.2 | 16.7 | 26.2 | **25** | 26.4 | 17.9 | **17.6** | 18.6 | 13.1 | 13.4 | 14.1 |
| ANDES | 362.8 | 362.8 | 386.9 | 332.1 | 334.9 | 355.7 | 250.3 | 250.1 | 256.9 | 156.3 | 156.3 | 157.7 | 159.9 | 159.4 | 158.8 | 78.3 | 78.1 | **76.4** |
| ASIA | 2.8 | 3.2 | 3.2 | 1.5 | 1.4 | 1.4 | 2.1 | 1.9 | 1.8 | 2.3 | 2.7 | 2.7 | 1.5 | 1.1 | 1.3 | 0.8 | 0.8 | 0.8 |
| BARLEY | 77.1 | 78.7 | 78.4 | 71.5 | 72.7 | **70.1** | 64.2 | 65.8 | 64 | 83.9 | 84.9 | 85.5 | 80.8 | 80 | **79.4** | 62.6 | **58.8** | 62.8 |
| CHILD | 17.4 | 17.3 | 17.3 | 15.8 | 15.7 | 15.7 | 12.1 | 9.8 | 9.9 | 16.5 | 16.5 | 16.5 | 15.8 | 15.5 | 15.3 | 14.2 | **12.9** | 13.2 |
| HAILFINDER | 58.6 | **57** | 61.9 | 49 | **47.4** | 57.7 | 41 | 41.1 | 56 | 64.4 | **62.7** | 67.8 | 48.5 | **47.5** | 56.3 | 34.3 | 34.2 | 45.7 |
| HEPAR2 | 135.2 | 136.4 | 136.2 | 117.5 | 118.8 | 117.1 | 90.4 | 88.9 | 88.4 | 108.7 | 108.7 | 109 | 95.3 | 94.6 | 94.8 | 67.9 | **65.9** | 67.4 |
| INSURANCE | 42.1 | 44.2 | 44.4 | 34.4 | 39 | 37.9 | 26.9 | 28.9 | 27.1 | 43.2 | 43.3 | 43.1 | 41.5 | 42.1 | 42.6 | 39.8 | **38.9** | 39.4 |
| WATER | 79.3 | 79.3 | 79.3 | 78.1 | 78.3 | 78.4 | 72.3 | 76.3 | 75.3 | 60.2 | 60.2 | 60.2 | 60.3 | 60.3 | 60.3 | 56.1 | 56.1 | 56.1 |
| WIN95PTS | 103.6 | 108.4 | 117.4 | 92.4 | 95.9 | 105 | 70.3 | 70.1 | 81.2 | 83.1 | 82.3 | 86.7 | 68.4 | **65.5** | 71.7 | 49.8 | 48.4 | 60.1 |

TABLE 3.19: The average BIC scores per sample of the DAG learned from the hybrid algorithm (SLL + hill-climbing), with the AND-rule("A"), **symC**("C") and **symG**("G"). "n.e." is the short name of "no extension".

| | PDAG | | | | | | | | | DAG | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Sample size | 500 | | | 1000 | | | 5000 | | | 500 | | | 1000 | | | 5000 | | |
| | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G |
| ALARM | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | -11.051 | -11.034 | -11.023 | -13.017 | **-12.906** | **-12.953** | -11.926 | -11.917 | **-11.907** | -11.1 | -11.099 | **-10.943** |
| ANDES | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | -433.878 | -433.878 | **-433.265** | -91.196 | -91.182 | **-90.923** | -93.749 | -93.747 | **-93.681** |
| ASIA | -2.34 | -2.319 | -2.328 | -2.299 | -2.297 | -2.296 | -2.338 | -2.319 | -2.319 | -2.341 | -2.338 | -2.338 | -2.298 | -2.295 | -2.296 | -2.259 | -2.259 | -2.259 |
| BARLEY | -77.705 | **-77.065** | -78.421 | -73.081 | n.e. | n.e. | n.e. | n.e. | n.e. | -74.424 | **-74.04** | **-74.148** | -70.415 | **-69.927** | **-70.272** | -61.45 | **-61.154** | **-61.248** |
| CHILD | -13.617 | -13.659 | -13.423 | -13.017 | -12.751 | -13.018 | -12.683 | -12.619 | **-12.65** | -13.369 | -13.369 | -13.369 | -12.986 | -12.985 | -12.985 | -12.534 | **-12.512** | **-12.515** |
| HAILFINDER | -82.419 | -80.01 | -82.828 | -81.498 | **-80.573** | -83.082 | -60.942 | -60.859 | -58.151 | -57.166 | **-57.104** | **-55.876** | -56.068 | **-56.005** | **-53.307** | -54.823 | -54.785 | **-50.537** |
| HEPAR2 | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | -33.643 | **-33.639** | **-33.639** | -33.278 | **-33.271** | **-33.271** | -32.798 | **-32.79** | **-32.789** |
| INSURANCE | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | -15.63 | **-15.523** | -15.615 | -14.819 | **-14.773** | **-14.788** | -13.846 | **-13.784** | **-13.8** |
| WATER | -28.637 | -28.637 | -28.637 | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | -14.283 | -14.283 | -14.283 | -13.736 | -13.736 | -13.736 | -13.164 | **-13.146** | -13.162 |
| WIN95PTS | -12.14 | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | -12.175 | **-11.897** | **-11.536** | -11.645 | **-11.322** | **-10.988** | -10.699 | **-10.517** | **-10.188** |

TABLE 3.20: The average BDeu scores per sample of the DAG learned from the hybrid algorithm (SLL + hill-climbing), with the AND-rule("A"), **symC**("C") and **symG**("G"). "n.e." is the short name of "no extension".

| | PDAG | | | | | | | | | DAG | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Sample size | 500 | | | 1000 | | | 5000 | | | 500 | | | 1000 | | | 5000 | | |
| | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G | A | C | G |
| ALARM | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | -10.94 | -10.919 | -10.894 | -12.348 | **-12.213** | **-12.287** | -11.506 | **-11.487** | **-11.451** | -11.01 | **-11.008** | **-10.849** |
| ANDES | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | -433.445 | -433.445 | **-432.834** | -90.996 | -90.982 | **-90.721** | -93.691 | -93.689 | **-93.625** |
| ASIA | -2.32 | -2.298 | -2.307 | -2.288 | -2.286 | -2.285 | -2.332 | -2.314 | -2.314 | -2.323 | -2.319 | -2.319 | -2.288 | -2.285 | -2.286 | -2.256 | -2.256 | -2.256 |
| BARLEY | -69.347 | **-68.791** | -69.917 | -66.369 | n.e. | n.e. | n.e. | n.e. | n.e. | -71.583 | **-71.07** | **-71.198** | -68.024 | **-67.032** | **-67.777** | -57.721 | **-57.225** | **-57.522** |
| CHILD | -13.496 | -13.515 | -13.436 | -13.033 | -12.762 | -13.041 | -12.677 | -12.622 | **-12.652** | -13.378 | -13.378 | -13.378 | -13.007 | -13.006 | -13.005 | -12.547 | **-12.526** | **-12.529** |
| HAILFINDER | -56.723 | **-56.417** | **-55.774** | -55.887 | -55.374 | **-53.694** | -54.757 | -54.681 | -51.071 | -55.837 | **-55.755** | **-54.607** | -55.318 | **-55.245** | **-52.691** | -54.637 | -54.598 | **-50.394** |
| HEPAR2 | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | -33.618 | **-33.613** | **-33.613** | -33.262 | **-33.255** | **-33.255** | -32.792 | **-32.784** | **-32.783** |
| INSURANCE | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | -15.211 | **-15.047** | -15.184 | -14.494 | **-14.439** | **-14.454** | -13.739 | **-13.669** | **-13.69** |
| WATER | -13.717 | -13.717 | -13.717 | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | -13.818 | -13.818 | -13.818 | -13.448 | -13.448 | -13.448 | -13.07 | **-13.048** | -13.068 |
| WIN95PTS | -11.785 | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | n.e. | -11.901 | **-11.6** | **-11.199** | -11.426 | **-11.081** | **-10.737** | -10.627 | **-10.434** | **-10.107** |

TABLE 3.21: The complexity of the **symG** and **symC**. The maximum number of nodes in the greedy search (**symG**) and maximum size of conditioning set(**symC**) for GS, IAMB, MMPC, and SI-HITON-PC with 5,000 samples.

| | maximum local search (**symG**) | | | | maximum conditioning set(**symC**) | | | |
|---|---|---|---|---|---|---|---|---|
| | GS | IAMB | MMPC | SI-HITON-PC | GS | IAMB | MMPC | SI-HITON-PC |
| ALARM | 9 | 9 | 11 | 9 | 3 | 2 | 4 | 4 |
| ANDES | 20 | 18 | 17 | 16 | 8 | 1 | 7 | 7 |
| ASIA | 6 | 6 | 6 | 6 | 1 | 1 | 3 | 2 |
| BARLEY | 7 | 7 | 8 | 7 | 1 | 1 | 2 | 2 |
| CHILD | 10 | 9 | 11 | 10 | 3 | 1 | 4 | 3 |
| DIABETES | 7 | 6 | 8 | 7 | 2 | 1 | 3 | 2 |
| HAILFINDER | 11 | 10 | 11 | 8 | 1 | 1 | 5 | 3 |
| LINK | 16 | 21 | 13 | 11 | 4 | 2 | 5 | 4 |
| MILDEW | 6 | 7 | 7 | 6 | 2 | 1 | 2 | 2 |
| HEPAR2 | 13 | 13 | 13 | 11 | 3 | 2 | 6 | 4 |
| INSURANCE | 8 | 11 | 9 | 10 | 3 | 3 | 4 | 4 |
| MUNIN1 | 8 | 9 | 9 | 11 | 3 | 1 | 4 | 4 |
| PATHFINDER | 6 | 7 | 8 | 9 | 1 | 1 | 2 | 2 |
| PIGS | 13 | 11 | 13 | 11 | 3 | 1 | 5 | 5 |
| WATER | 6 | 6 | 8 | 6 | 1 | 1 | 3 | 2 |
| WIN95PTS | 13 | 14 | 17 | 14 | 4 | 3 | 6 | 6 |

TABLE 3.22: The average running time of symmetry correction (in second) for GS, IAMB, MMPC, and SI-HITON-PC, using **symG** and **symC** with 5,000 samples. The experiments were performed on Intel Xeon E5-1650 Processor(Six Core HT, 3.2GHz Turbo) with 16GB of RAM.

| | **symG** | | | | **symC** | | | |
|---|---|---|---|---|---|---|---|---|
| | GS | IAMB | MMPC | SI-HITON-PC | GS | IAMB | MMPC | SI-HITON-PC |
| ALARM | 0.61 | 0.08 | 0.54 | 0.05 | 1.74 | 0.1 | 0.61 | 0.06 |
| ANDES | 12.25 | 0.84 | 36.73 | 2.48 | 14.14 | 0.81 | 5.54 | 0.51 |
| ASIA | 0.08 | 0.01 | 0.06 | 0.01 | 0.09 | 0.01 | 0.03 | 0.01 |
| BARLEY | 0.89 | 0.86 | 0.62 | 0.17 | 1.13 | 0.12 | 0.69 | 0.09 |
| CHILD | 0.55 | 0.07 | 0.49 | 0.06 | 1.03 | 0.05 | 0.24 | 0.03 |
| DIABETES | 4.81 | 1.31 | 7.28 | 2.45 | 13.23 | 1.92 | 11.23 | 1.81 |
| HAILFINDER | 1.74 | 0.53 | 0.84 | 0.17 | 1.77 | 0.15 | 0.49 | 0.08 |
| LINK | 40.3 | 9.81 | 18.46 | 1.62 | 41.17 | 4.38 | 16.22 | 2.72 |
| MILDEW | 0.19 | 0.07 | 0.31 | 0.09 | 0.4 | 0.06 | 0.34 | 0.06 |
| HEPAR2 | 2.83 | 0.26 | 4.5 | 0.37 | 2.69 | 0.2 | 1.45 | 0.16 |
| INSURANCE | 0.41 | 0.06 | 0.44 | 0.05 | 1.25 | 0.09 | 0.57 | 0.06 |
| MUNIN1 | 3.5 | 0.54 | 1.55 | 0.29 | 6.06 | 0.61 | 5.43 | 0.51 |
| PATHFINDER | 1.16 | 0.21 | 1.61 | 0.27 | 2.26 | 0.23 | 1.38 | 0.25 |
| PIGS | 40.96 | 18.42 | 7.61 | 1.72 | 35.62 | 2.13 | 5.53 | 0.8 |
| WATER | 0.1 | 0.03 | 0.12 | 0.03 | 0.38 | 0.04 | 0.12 | 0.02 |
| WIN95PTS | 3.82 | 0.23 | 3.93 | 0.22 | 3.21 | 0.2 | 1.58 | 0.13 |

# Chapter 4

# Weighted MAX-SAT for V-structure Selection in Bayesian Network Structure Learning

After obtaining the skeleton, the next key step is to orient the directions of the edges in the graph. The quality of the final output network highly depends on the edge orientation. Detection of V-structures is important in the edge orientation as V-structures carry the conditional dependency relation between nodes. Previously the V-structures are first learned based on the conditional independence tests and the significant ones are applied to the skeleton greedily one by one based on their p-values. To avoid the conflicts, the later applied V-structure which conflicts with any previously applied V-structure is ignored. For V-structure selection, the aforementioned greedy method is straightforward but it does not consider the relationship among V-structures. This motivates us to develop a method that can consider all V-structures jointly as a few ignored V-structures may together beat a more significant previously applied V-structure. Using data to test the V-structure is an example of multiple uses of data in structure learning. Furthermore, when we consider all V-structures jointly, for a particular V-structure, the data in other V-structure helps to determine its existence. This also can be viewed as a kind of multiple uses of data through network connections. To represent the confidence or strength of V-structures, we propose two methods to compute their weights by using the information of conditional dependence and dependence. We overcome

the limitation of the greedy method by applying weighted maximum satisfiability (MAX-SAT) to deal with all possible V-structures together and thus obtain a better solution.

This chapter is organized as follows: we first briefly introduce the weighted MAX-SAT, which serves as a formulation of our problem in Section 4.1; we then describe the edge orientation problem and the greedy method for edge orientation in Section 4.2; we propose our edge orientation approach, including two weighting methods and a weighted MAX-SAT formulation in Section 4.3, followed by the empirical evaluation in Section 4.4. Finally, we summarize this chapter in Section 4.5.

## 4.1 Weighted MAX-SAT

Weighted MAX-SAT is a variant of the SAT problem, where weights are added to each clause and the goal is to find an assignment that maximises the sum of the weights of satisfied clauses. To solve the SAT problem or the weighted MAX-SAT problem, one approach is to search for a satisfying assignment of a *conjunctive normal form* (CNF) formula by flipping the truth-values of atoms in that CNF. Common choices of flipping include flips that decrease the number or weights of unsatisfied clauses and random flips. One of the successful algorithms called the local search WalkSAT algorithm [41] combines these two kinds of flips and achieves good results.

Note that weighted MAX-SAT has been used in Bayesian network learning in [42], which encodes the total order or ancestors of all nodes into clauses with weights transformed from the corresponding scores. The difference between our proposed approach and previous ones is that we apply weighted MAX-SAT to V-structures which are more complicated with essential relations among three nodes in BNs rather than the parent-child relations.

## 4.2 Edge Orientation in Bayesian Network Structure Learning

Most of the constraint-based BN structure learning algorithms consist of three stages [13]:

1. Learning the skeleton of the network.
2. Setting the directions of arcs in the detected V-structures.
3. Setting the directions of other arcs while satisfying the acyclicity constraint.

We focus on the detection (selection) of V-structures in the second stage. For the constraint-based algorithms, the V-structures are usually detected by conditional independence tests. For $X \rightarrow Z \leftarrow Y$, the test checks if $X$ and $Y$ are dependent given $Z$ and shows its significance in a p-value. Furthermore, the significant V-structures are selected greedily by sorting with increasing p-values within a certain threshold $\alpha$ (e.g. $\alpha = 0.05$). This means that a more significant V-structure is put front and a greedy method is applied to this sorted list to apply V-structures to the skeleton one by one. If a later V-structure contradicts any previously applied V-structure in terms of a conflict in the edge orientation or producing a directed cycle, then it is ignored and the algorithm continues to apply V-structures until all significant V-structures are processed.

On the other hand, the score-based algorithms directly find the DAG by searching, so they do not need to consider the edge orientation alone. The constrained hill-climbing search used in the hybrid algorithms is similar to the constraint-based algorithms because it requires the skeleton, which is used to restrict the search. The greedy method in the constraint-based algorithms and the constrained hill-climbing search used in the hybrid algorithms serve as two baseline methods for our method.

Our approach is to replace the greedy method in the constraint-based algorithms of finding V-structures with the weighted MAX-SAT solver. We consider all possible V-structures as a whole and test them together. In contrast to testing each V-structure separately, our approach has a few advantages. First, a single test is

prone to be wrong and limited by the amount of data used. A significant wrongly found V-structure can dominate and worsen the structure learning as shown in the following example. Instead, our approach allows the selection of V-structures to be optimised by considering their weights. Second, the significance level of the tests affects the test results and the selection of V-structure. Thus, the significance level can be very sensitive to the learned structure. Instead, our approach avoids setting the significance level and considers all V-structures, including previously insignificant V-structures by computing relatively low weights for them. Our idea shares some similarities with the edge orientation as constraint optimization in [43], which considers the constraints jointly. But they did not extract explicit weights for the constraints.

As an illustrative example shown in Figure 4.1, in the "W" shape undirected graph in Figure a, there are three possible V-structures: $vs(A, D, B)$, $vs(D, B, E)$ and $vs(B, E, C)$. $vs(D, B, E)$ conflicts with $vs(A, D, B)$ and $vs(B, E, C)$ respectively as they contains reversed arcs. Suppose $vs(D, B, E)$ is most significant with the least p-value, but the DAG containing $vs(A, D, B)$ and $vs(B, E, C)$ (Figure f) is the true network. The greedy method will output Figure c. However, our approach can output Figure f after considering the weights of all three V-structures with weighted MAX-SAT.

## 4.3 Proposed Methodology

We describe our approach in two parts: learning the weights of V-structures and solving the V-structures selection problem with weighted MAX-SAT. In the first part, we introduce two methods to compute weights for each V-structure from data. In the second part, the learned weights and their corresponding V-structures are imported into the weighted MAX-SAT solver with constraints, which are extracted from the skeleton. The solution decides what V-structures should be selected. The selected V-structures together with the skeleton can give a partial directed acyclic graph (PDAG), which may contain both directed and undirected edges.

From Theorem 2.1.1, we know that the skeleton and the V-structures together uniquely decide the equivalence class a BN belongs to. Suppose the learned skeleton is perfect. A key issue becomes how to accurately find the true V-structures

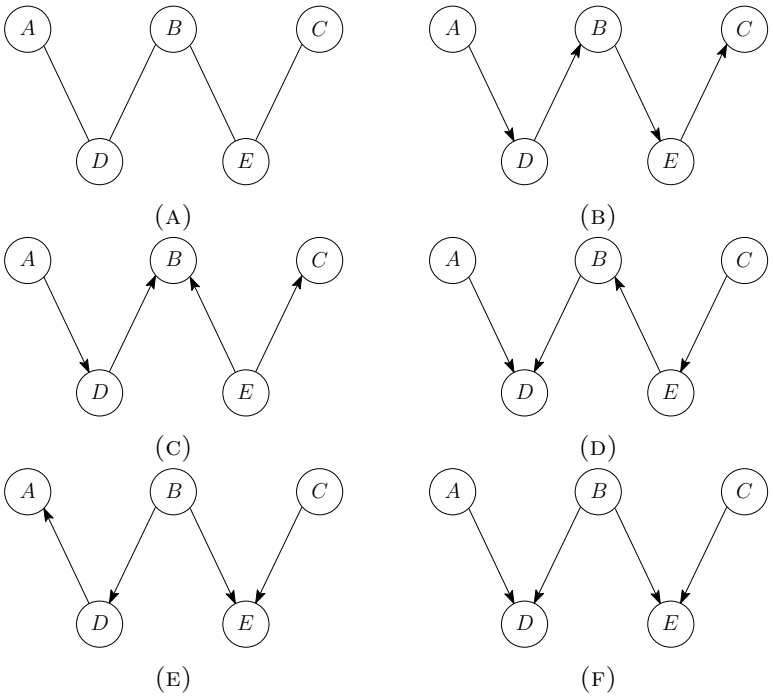



FIGURE 4.1: An example of possible DAGs with different V-structures. (a) The skeleton of the DAGs. (b) A DAG with no V-structure. (c) A DAG with the V-structure $D \rightarrow B \leftarrow E$. (d) A DAG with the V-structure $A \rightarrow D \leftarrow B$. (e) A DAG with the V-structure $B \rightarrow E \leftarrow C$. (f) The DAG with the V-structures $A \rightarrow D \leftarrow B$ and $B \rightarrow E \leftarrow C$.

from the data. In the max-min hill-climbing algorithm (MMHC) [20], a hybrid structure learning algorithm, the search of edge orientations takes 10 times as skeleton learning [6], which can be viewed as an indirect search of V-structures.

Our weighting methods measure the strength of V-structures. Similarly, to orient the edges, some researchers attempted to measure the reliability of independence by estimating the probabilities of the logic statements [44]. However, the statements are selected based on the threshold and sorted. More reliable statements are kept first as the greedy method selects more significant V-structures first. They have the same problem of applying wrong dominating V-structures or statements as shown in our illustrative example in Figure 4.1. Our weighted MAX-SAT formulation can overcome this problem.

Our approach optimises the selection of V-structures while another closely related work weighs the conditional independence constraints and minimises the total weight of the violated constraints [45]. The difference is that we only consider V-structures as key structural dependence constraints while they consider all dependence (independence) constraints. As an exact search method, they find the

optimal graph. But the largest networks in their experiments contains only 12 nodes, even if they already provide efficient ways to check and update the constraints. On the other hand, our methods consider V-structures, a small number of dependence constraints on the skeleton. This allows us to capture the dependence information to orient edges and handle relatively large graphs.

### 4.3.1 Learning Weights of V-structures

Since a V-structure $vs(X_1, Z, X_2)$ contains two pieces of information, the conditional dependence $X_1 \not\perp\!\!\!\perp X_2|Z$ and the independence $X_1 \perp\!\!\!\perp X_2$, the weight that measures the strength of the V-structure should cover both of them. In a V-structure, it is assumed that the parents $X_1$ and $X_2$ are independent so the criterion for the V-structure detection is the conditional dependence $X_1 \not\perp\!\!\!\perp X_2|Z$. However, in the full graph of a DAG, the independence $X_1 \perp\!\!\!\perp X_2$ may not hold due to a possible common cause of $X_1$ and $X_2$ or a possible directed path from one of $X_1$ and $X_2$ to the other. So we need to consider this dependence between $X_1$ and $X_2$ in our weights. We choose to use the additional dependence by knowing $Z$, i.e., the difference between the conditional dependence $X_1 \not\perp\!\!\!\perp X_2|Z$ and the dependence $X_1 \not\perp\!\!\!\perp X_2$, to measure the strength of V-structure.

With the independence $X_1 \perp\!\!\!\perp X_2$ assumption, the strength of V-structure can be measured by the conditional dependence $X_1 \not\perp\!\!\!\perp X_2|Z$ alone, because the additional dependence by knowing $Z$ is equal to the conditional dependence. In a DAG, without the independence assumption, the additional dependence $X_1 \not\perp\!\!\!\perp X_2|Z$ by knowing $Z$ becomes the difference between the conditional dependence $X_1 \not\perp\!\!\!\perp X_2|Z$ and the dependence $X_1 \not\perp\!\!\!\perp X_2$. Suppose $X_1$ and $X_2$ are highly dependent due to a common cause, then a high conditional dependence $X_1 \not\perp\!\!\!\perp X_2|Z$ does not necessarily indicate a V-structure as the dependence is not due to the conditioned $Z$. On the other hand, suppose $X_1$ and $X_2$ are not very dependent by conditioning $Z$, but $X_1$ and $X_2$ are not dependent at all. This could still suggest a possible V-structure. The additional dependence measures the dependence from conditioning on $Z$ by taking the direct dependence $X_1 \not\perp\!\!\!\perp X_2$ into consideration as a baseline.

So we consider two cases: (1) the conditional dependence $X_1 \not\perp\!\!\!\perp X_2|Z$ and (2) the difference between the conditional dependence $X_1 \not\perp\!\!\!\perp X_2|Z$ and the dependence $X_1 \not\perp\!\!\!\perp X_2$.

We propose two methods to estimate weights for V-structures. The most straight forward one is to use statistics obtained in conditional independence (CI) tests. The second one is based on the Bayes factor to compare the likelihood of the existence of V-structure and its absence (the conditionally independent model).

**Mutual Information** To measure statistical independence between variables, in this chapter we adopt Mutual Information (MI) or conditional Mutual Information (CMI), where $\mathrm{MI}(X_1, X_2)$ and $\mathrm{CMI}(X_1, X_2|Z)$ are non-negative and $\mathrm{MI}(X_1, X_2) = 0$ ($\mathrm{CMI}(X_1, X_2|Z) = 0$) if and only if $X_1$ and $X_2$ are independent (conditionally independent given $Z$). MI between variables $X_1$ and $X_2$ measures the amount of information shared by those two variables, which is defined as follows [46]:

$$\mathrm{MI}(X_1, X_2) = \sum_{x_1, x_2} P(x_1, x_2) \log \left( \frac{P(x_1, x_2)}{P(x_1)P(x_2)} \right).$$

Similarly, CMI between variables $X$ and $Y$, conditioning on $Z$ is defined as follows.

$$\mathrm{CMI}(X_1, X_2|Z) = \sum_{x_1, x_2, z} P(x_1, x_2|z) \log \left( \frac{P(x_1, x_2|z)}{P(x_1|z)P(x_2|z)} \right).$$

In our case on V-structures, a high value of CMI indicates a dependency between parents $X_1$ and $X_2$, conditioning on the common child $Z$ while a high value of MI indicates a dependency between parents $X_1$ and $X_2$. Both CMI and the difference between CMI and MI could imply the existence of the V-structure $vs(X_1, Z, X_2)$. Hence, besides only using CMI as the weights, we also consider the difference between CMI and MI as the weights.

**Bayes Factors** Bayes factor is a ratio of likelihood between different hypothesis models [47]: $\frac{P(d|g_1)}{P(d|g_2)}$, where $d$ is the data, $g_1$ is the dependent model and $g_2$ is the independent model. It has been used in conditional independence (CI) tests to decide BNs structures [48], encoded into the recursive autonomy identification (RAI) algorithm [49] to learn BNs structures [50].

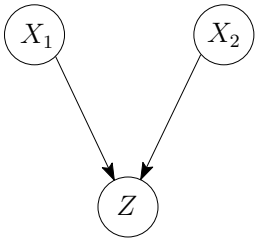
The marginal likelihood of a Bayesian network of $n$ discrete variables $\{x_1, \ldots, x_n\}$ with Dirichlet priors can be expressed as follows, which is known as *Bayesian*

*Dirichlet* (BD) score [36]:

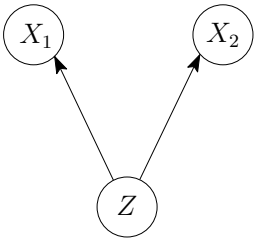
$$P(d|g,\alpha) = \prod_{i=1}^{n}\prod_{j=1}^{q_i}\frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij}+c_{ij})}\prod_{k=1}^{r_i}\frac{\Gamma(\alpha_{ijk}+c_{ijk})}{\Gamma(\alpha_{ijk})},$$

where $c_{ijk}$ represents the number of examples of $X_i = k$ when its parent set $\Pi_i = j$, $c_{ij} = \sum_{k=1}^{r_i} c_{ijk}$, $\alpha_{ijk}$ is the hyperparameters of the Dirichlet priors, $\alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}$, and $q_i$ signifies the number of instances of $\Pi_i$, $q_i = \prod_{x_l \in \Pi_i} r_l$. $\Gamma(x)$ is the gamma function[1]. When $\alpha_{ijk} = 1$, BD becomes the K2 score [51]; when $\alpha_{ijk} = 1/2$, BD becomes the BD score with Jeffreys' prior [52]; and when $\alpha_{ijk} = \bar{\alpha}/(r_i q_i)$, where $\bar{\alpha}$ is known as the *imaginary* or *equivalent sample size*, BD becomes the *Bayesian Dirichlet equivalent uniform* (BDeu) score [36].

We modify the hypothesis models to represent the existence and the absence of a V-structure using Bayes factor. The difference is that we compute Bayes factor for a V-structure after learning the skeleton while methods in [48, 50] update the learned DAG, including its underlying skeleton structure based on the Bayes factor. Specifically, we compute two Bayes factors for one V-structure: $B_1 = \frac{P(d|u_1)}{P(d|u_2)}$ for conditional dependence and $B_2 = \frac{P(d|h_1)}{P(d|h_2)}$ for dependence, as shown in Figures 4.2 and 4.3, respectively. After that we take logarithm on them to generate weights: $\log(B_1)$ if we only consider the conditional dependence and $\log(B_1) - \log(B_2)$ if we consider the difference between the conditional dependence and the dependence.




(A) $u_1$; conditionally dependent.




(B) $u_2$; conditionally independent.

FIGURE 4.2: Hypothesis models for a V-structure.


(A) $h_1$; dependent model.


(B) $h_2$; independent model.

FIGURE 4.3: Hypothesis models for the parents in a V-structure.

[1]The gamma function is defined as $\Gamma(x) = \int_0^\infty s^{x-1}e^{-t}dt$. When $x$ is a non-negative integer, $\Gamma(x+1) = x!$.

### 4.3.2 Weighted MAX-SAT Formulation

We now describe how to formulate V-structure selection as a weighted MAX-SAT problem.

**Unit Clauses** Let a Boolean variable $V_i$ represent the existence of the $i$-th V-structure. The unit clauses are $v_i$ and $\neg v_i$ with their correspondingly computed weights. For the method of mutual information, as CMI and MI are both non-negative, in order to avoid negative weights, CMI is assigned to $v_i$ while MI is assigned to $\neg v_i$. For the method of Bayes factor, if the weight $w$ is positive then it is assigned to $v_i$ and the weight of $\neg v_i$ is set to zero. If $w$ is negative then $|w|$ is assigned to $\neg v_i$ and the weight of $v_i$ is set to zero.

**Constraints** We define four kinds of constraints. The first three are hard constraints while the last one is soft and optional. The unit clauses are soft constraints as well. The hard constraints are assigned with a weight equal to the sum of the weights of soft constraints.

- **Conflict of V-structures:** If one V-structure contains an arc and another V-structure contains the reversed arc of that arc, then there is a conflict between them. This means that the two V-structures cannot exist at the same time in the graph. Let the $j$-th and $k$-th V-structures be such two V-structures. We define a clause as follows,

$$\neg v_j \vee \neg v_k. \tag{4.1}$$

- **V-structures implication:** A V-structure contains two directed arcs. Two V-structures may share an arc if they share the child node. When we confirm the existence of some V-structures, this may automatically imply some other V-structures. For example, suppose there are four nodes $A, B, C$ and $X$ in a graph, where $A, B, C$ are disconnected and three edges are connecting $A, B$ and $C$ to $X$ respectively. If V-structures $(A \to X \leftarrow B)$ and $(B \to X \leftarrow C)$ are confirmed, then the V-structure $(A \to X \leftarrow C)$ is implied automatically. Therefore, we have $v_i \leftarrow (\vee_{j \in A_1} v_j) \wedge (\vee_{k \in A_2} v_k)$, where $A_1$ is the set of indices of V-structures which contain the first arc of the V-structure $v_i$ and similarly

$A_2$ is for the second arc. This can be converted to CNF equivalently as follows,

$$\wedge_{j\in A_1,k\in A_2}\ (v_i \vee \neg v_j \vee \neg v_k). \tag{4.2}$$

As the original constraint is a hard constraint, the converted clauses in the CNF are hard constraints as well.

- **Acyclicity:** By definition of DAG, there should be no directed cycles. The arcs of a set of V-structure may form a direct cycle. Thus we need constraints to avoid such a set of V-structures. Besides, V-structures can form directed cycles of different lengths. So we need to find all possible cycles in the skeleton and encode them into clauses. However, the number of cycles in BNs of only medium sizes can be huge. For each cycle, the number of generated constraints can be also large, because each directed arc in the cycle may be contained by a few V-structures. Let a directed cycle be of size $n$, and the number of V-structures containing the $i$-th arc be $m_i$. Then the number of constraints is $\prod_{i=1}^{n} m_i$. Due to the large number of constraints, we do not impose the acyclicity constraints but make the learned V-structures from the weighted MAX-SAT solver to be applied greedily. This means that when applying V-structures to the skeleton, the ones that produce a directed cycle will be ignored.

- **The penalty/gain on V-structures sharing child:** The intuition on adding a penalty on the V-structures sharing the same child is two-fold. The first one is to regularize the number of parameters in the learned BN. This is similar to the regularization used in the *Bayesian Information Criterion* (BIC). The second one is to accommodate the additional weight gains due to implied V-structures. Suppose a few disconnected parents are sharing the same child forming some V-structures, and for each possible V-structure, the weight $w_i$ for $v_i$ is greater than the weight $\hat{w}_i$ for $\neg v_i$. Then an additional confirmed V-structure on the same child will produce not only a gain from its own weight but also gains from many implied new V-structures. A shared child is contained in more V-structures, then it is more attractive to new V-structures. For the $j$-th and the $k$-th V-structures sharing the same child, we can define

$$\neg v_j \vee \neg v_k. \tag{4.3}$$

The clause is not satisfied when both $v_j$ and $v_k$ are TRUE. The weight can be set according to the weight differences of $v_j$ and $v_k$ against their own negation. One intuition behind this weight setting is that the direct gain obtained from violating this clause is the minimum of the weights of $v_j$ and $v_k$. Thus, the weight can be set to this direct gain to offset additional gain effect due to the implied new V-structures. On the other hand, if we have some domain knowledge about the dataset, which tells us that there exist many V-structures sharing the same child, then constraints can be created to encourage such structures as follows,

$$v_j \wedge v_k. \tag{4.4}$$

However, as some of our experimental results get worse when adding this kind of constraint, we omit it.

In summary, we only impose the conflict and implication constraints. We feed those constraints and the unit clauses into MAX-SAT solver. The solution is matched with corresponding V-structures. These V-structures are applied to the skeleton greedily to avoid directed cycles.

### 4.3.3 Algorithm of V-structure selection through Weighted MAX-SAT

In this section, we present our complete algorithm for V-structure selection via weighted MAX-SAT. In Algorithm 5, we input the skeleton and the dataset and choose a weighting method. We first extract all possible V-structures from the skeleton and compute the weights for their existence and absence (lines 1-2). Then we extract the clauses from the conflict and implication constraints, which are obtained from the arcs of V-structures (line 3). The weighted MAX-SAT solver is used to find the selected V-structures, given the clauses with their weights (line 4). At last, we apply the selected V-structures greedily to the skeleton to avoid cycles and obtain our PDAG (line 5). The outputs are the selected V-structures and the PDAG.

**Algorithm 5:** V-structure Selection through Weighted MAX-SAT

**Input:** The skeleton $\mathcal{SK}$, a weighting method $\mathcal{W}$, dataset $\mathcal{D}$

**Output:** A set of V-structures $\mathcal{VS}_{selected}$ obtained from weighted MAX-SAT solver and a partial DAG $\mathcal{G}$ which contains V-structures in $\mathcal{VS}_{selected}$.

1: Extract all possible V-structures from $\mathcal{SK}$ and let $\mathcal{VS}_{all} = \{V_i\}_{i=1}^{n}$.
2: Compute the weights $w_i$ and $\hat{w}_i$ for the corresponding Boolean variable $v_i$ and its negation $\neg v_i$ of each V-structure in $\mathcal{VS}_{all}$ using method $\mathcal{W}$.
3: Based on the arcs of V-structures in $\mathcal{VS}_{all}$, produce the conflict and implication hard constraints as clauses.
4: Feed all clauses with their weights into the weighted MAX-SAT solver and obtain the set of selected V-structures $\mathcal{VS}_{selected}$ as the solution.
5: Apply $\mathcal{VS}_{selected}$ to the skeleton $\mathcal{SK}$ greedily to avoid directed cycles and obtain the the partial DAG $\mathcal{G}$.
6: **return** $\mathcal{VS}_{selected}$ and $\mathcal{G}$.

TABLE 4.1: Information of the BNs used in the experiments.

| Name | Nodes | Arcs | Parameters | Maximum in-degree |
|---|---|---|---|---|
| ALARM | 37 | 46 | 509 | 4 |
| ANDES | 223 | 338 | 1157 | 6 |
| ASIA | 8 | 8 | 18 | 2 |
| CHILD | 20 | 25 | 230 | 2 |
| MILDEW | 35 | 46 | 540,150 | 3 |
| INSURANCE | 27 | 52 | 984 | 3 |
| WIN95PTS | 76 | 112 | 574 | 7 |

## 4.4 Experiments

We evaluate our proposed weighted MAX-SAT method, and two baseline methods: the greedy method with conditional independence tests and the constrained hill-climbing search used in the hybrid algorithms, on both true skeletons and learned skeletons, using datasets of size $10^2, 10^3$ and $10^4$ sampled from the probability distributions of 7 BNs as shown in Table 4.1 from the BN repository of **bnlearn** [2] [13]. The tests used in the greedy method are the default tests in the **bnlearn** package, which are the asymptotic mutual information with $\alpha = 0.05$.

For our MAX-SAT method, we attempt many weighting methods and their variations. One major variation is whether we only consider the conditional dependence of the two parent nodes in the V-structure or include their dependence to compute the dependence difference. For methods using the Dirichlet priors, we run three kinds of hyperparameters ($\alpha_{ijk} = 1$ (k2); $\alpha_{ijk} = 1/2$ (Jeffreys), and $\alpha_{ijk} = \bar{\alpha}/(r_i q_i)$

[2] http://www.bnlearn.com/bnrepository/

with $\alpha=1$ (BDeu)). We use Open-WBO [53] with default settings as the weighted MAX-SAT solver on Intel Xeon E5-1650 Processor(Six Core HT, 3.2GHz Turbo) with 16GB of RAM. There are 1,032 variables, 2,064 soft clauses and 20,541 hard clauses in the weighted MAX-SAT problem for the true skeleton of the largest BN we used, ANDES. And this does not change as the sample size rises. It is much smaller than the weighted MAX-SAT in [42]. The greedy method is included in the constraint-based algorithms in **bnlearn**. Their corresponding hybrid algorithms are also implemented in **bnlearn** [20] with the true skeleton and the default score criterion, BIC.

The metrics of performance are Structural Hamming Distance (SHD) [20], precision and recall of the learned V-structures compared to the V-structures in the true network. A low SHD indicates that the learned network is close to the true network which is preferred. A high precision or recall indicates a better match to the V-structures in the true network. Sometimes the algorithm does not output any V-structure. For instance, in the greedy method if no V-structure is tested to be significant then it will not output any V-structure. In this case, we set their precision to zero and their recall is also zero. We repeat the experiments with different samples generated from the true network with different seeds for 10 times and take the averaged SHD, precision and recall.

TABLE 4.2: Performance in averaged SHD on true skeletons. "OT" means "out of time", "d" means "difference between conditional dependence and dependence", "bf" means "Bayes factors", and "k2" denotes the hyperparameter $\alpha_{ijk} = 1$.

| | ALARM | | | ANDES | | | ASIA | | | CHILD | | | MILDEW | | | INSURANCE | | | WIN95PTS | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| sample size | $10^2$ | $10^3$ | $10^4$ | $10^2$ | $10^3$ | $10^4$ | $10^2$ | $10^3$ | $10^4$ | $10^2$ | $10^3$ | $10^4$ | $10^2$ | $10^3$ | $10^4$ | $10^2$ | $10^3$ | $10^4$ | $10^2$ | $10^3$ | $10^4$ |
| greedy | 34.7 | 22.6 | 14.3 | 156.8 | 74.8 | 41 | 4.6 | 0.3 | 1.3 | 13.9 | 12.1 | 12.1 | 46 | 21 | 18.5 | 32.6 | 28.2 | 29.2 | 72.2 | 40.3 | 32.7 |
| search | 39.1 | 16.8 | 12.9 | 152.2 | 68.3 | 34.1 | 4 | 0.9 | 0.8 | 18.4 | 14.5 | 12.2 | 46 | 46 | 41 | 44.5 | 41.7 | 39.7 | 76.3 | 33.6 | 23.5 |
| GOBNILP | 12.2 | 3.7 | 1.8 | OT | | | 2.3 | 0.7 | 0.2 | 16.6 | 9 | 0 | 39.5 | 30.9 | 16.6 | 31.4 | 20.4 | 11.8 | OT | | |
| cmi | 26.3 | 23.7 | 21 | 94.9 | 31.3 | 13.2 | 3.2 | 2.7 | 1.8 | 18.4 | 19.6 | 20 | 21.8 | 14 | 12 | 36.5 | 40.1 | 41 | 63.5 | 45.4 | 43.8 |
| cmi-d | 8 | 1.6 | 1.2 | 68.7 | 15.3 | 0.6 | 1.9 | 0.2 | 0.2 | 14.7 | 12.3 | 10.4 | 14.6 | 9.3 | 1 | 32.3 | 21.4 | 22 | 33.7 | 11.2 | 9.3 |
| bf_k2 | 11.9 | 4.7 | 2 | 101.3 | 39.6 | 13.2 | 1.7 | 0.2 | 0.2 | 13.4 | 10.3 | 5.7 | 25.7 | 24.7 | 17 | 36.9 | 29.2 | 26.8 | 19.8 | 12.4 | 8.6 |
| bf_k2-d | 13 | 3.5 | 3 | 111.9 | 55.8 | 26.3 | 1.6 | 0.6 | 0.2 | 15.1 | 12.7 | 9.6 | 23.4 | 21.7 | 17 | 34.6 | 28.8 | 26.7 | 17.1 | 13.1 | 9.8 |

### 4.4.1 Experimental Results

Our experiments are run on true skeletons and learned skeletons. Experiments on true skeletons are used to show the full potential of the orientation methods. By

TABLE 4.3: Performance in averaged SHD using Bayes factors with different hyperparameters on true skeletons. "d" means "difference between conditional dependence and dependence", "bf" means "Bayes factors", and "Jeffreys" and "BDeu" denote the hyperparameter $\alpha_{ijk}$ taking the values of 1/2 and $1/(r_i q_i)$, respectively.

| | ALARM | | | ANDES | | | ASIA | | | CHILD | | | MILDEW | | | INSURANCE | | | WIN95PTS | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| sample size | $10^2$ | $10^3$ | $10^4$ | $10^2$ | $10^3$ | $10^4$ | $10^2$ | $10^3$ | $10^4$ | $10^2$ | $10^3$ | $10^4$ | $10^2$ | $10^3$ | $10^4$ | $10^2$ | $10^3$ | $10^4$ | $10^2$ | $10^3$ | $10^4$ |
| bf_Jeffreys-d | 37.1 | 13.6 | 4.7 | 181.8 | 61.7 | 39 | 5 | 1.6 | 0 | 11.6 | 10.3 | 9.9 | 24.5 | 19.4 | 17.4 | 33.5 | 25 | 22.7 | 73.7 | 48.3 | 20.9 |
| bf_BDeu-d | 12 | 3 | 2.5 | OT | 27.6 | 15 | 1.9 | 0.6 | 0.2 | 16.8 | 11.9 | 8.1 | 22.3 | 19.2 | 15 | 22.4 | 23.8 | 18.8 | 28.8 | 10.3 | 9.1 |

using true skeletons, we avoid the effect of the inaccurate skeletons and the true DAG can be learned in theory. So we could better compare the performances of orientation methods. In practice, we usually only have learned skeletons so we run experiments on learned skeletons to show our orientation methods work on inaccurate skeletons as well.

**On the True Skeletons.**

In Table 4.2, we show the results from the greedy method, search in the hybrid algorithms, the exact search algorithm, GOBNILP [3] [10] and our weighted MAX-SAT methods in terms of SHD. The first trend is that the SHD values drop as the amount of data examples increases for most cases. GOBNILP is given the true skeleton by removing inconsistent scores in the file. GOBNILP finds the optimal network and shows good performance but it cannot handle networks with a large number of variables (ANDES and WIN95PTS). For the weighting methods using MI, the additional dependence show much better performance. For the weighting methods using the Bayes factor, the additional dependence does not make a big difference. Here we take "k2" ($\alpha_{ijk} = 1$) as an example. Across seven datasets, "cmi-d" gives the best SHD for most cases. In Table 4.3, we show the results of the Bayes factor for the other two kinds of hyperparameters with the additional dependence. There are some performance differences but none of them outperforms "cmi-d" generally. It is worth noting that when the sample size is small ($10^2$), for some weighting methods, the solver needs a long time to finish or cannot finish (over 4 hours) for ANDES. But with a larger sample size ($10^3$), it only takes seconds. This could be due to the inconsistent weights computed from the insufficient data.

[3] https://www.cs.york.ac.uk/aig/sw/gobnilp/

TABLE 4.4: Performance in averaged precision on true skeletons. "d" means "difference between conditional dependence and dependence". "bf" means "Bayes factors".

| | ALARM | | | ANDES | | | ASIA | | | CHILD | | | MILDEW | | | INSURANCE | | | WIN95PTS | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| sample size | $10^2$ | $10^3$ | $10^4$ | $10^2$ | $10^3$ | $10^4$ | $10^2$ | $10^3$ | $10^4$ | $10^2$ | $10^3$ | $10^4$ | $10^2$ | $10^3$ | $10^4$ | $10^2$ | $10^3$ | $10^4$ | $10^2$ | $10^3$ | $10^4$ |
| greedy | 0.23 | 0.61 | 0.82 | 0.53 | 0.86 | 0.95 | 0.2 | 0.92 | 0.66 | 0.1 | 0.45 | 0.59 | 0 | 0.43 | 0.94 | 0.09 | 0.28 | 0.91 | 0.38 | 0.87 | 0.93 |
| search | 0.98 | 1 | 0.98 | 1 | 1 | 1 | 0.65 | 1 | 1 | 0 | 0.52 | 0.66 | 0 | 0 | 1 | 0.4 | 0.53 | 0.48 | 0.95 | 0.94 | 0.95 |
| GOBNILP | 0.92 | 1 | 1 | | OT | | 0.9 | 1 | 1 | 0.72 | 0.78 | 1 | 0.74 | 1 | 1 | 0.7 | 0.79 | 0.84 | | OT | |
| cmi-d | 0.93 | 1 | 1 | 0.66 | 0.89 | 1 | 0.75 | 0.97 | 0.97 | 0.13 | 0.3 | 0.68 | 0.52 | 0.61 | 0.96 | 0.13 | 0.5 | 0.61 | 0.84 | 0.97 | 0.97 |
| bf_k2-d | 0.63 | 0.95 | 0.95 | 0.41 | 0.68 | 0.84 | 0.77 | 0.9 | 0.97 | 0.25 | 0.56 | 0.83 | 0.37 | 0.51 | 0.72 | 0.13 | 0.25 | 0.37 | 0.81 | 0.94 | 0.97 |

TABLE 4.5: Performance in averaged recall on true skeletons. "d" means "difference between conditional dependence and dependence". "bf" means "Bayes factors".

| | ALARM | | | ANDES | | | ASIA | | | CHILD | | | MILDEW | | | INSURANCE | | | WIN95PTS | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| sample size | $10^2$ | $10^3$ | $10^4$ | $10^2$ | $10^3$ | $10^4$ | $10^2$ | $10^3$ | $10^4$ | $10^2$ | $10^3$ | $10^4$ | $10^2$ | $10^3$ | $10^4$ | $10^2$ | $10^3$ | $10^4$ | $10^2$ | $10^3$ | $10^4$ |
| greedy | 0.04 | 0.27 | 0.45 | 0.18 | 0.52 | 0.7 | 0.1 | 0.95 | 0.8 | 0.1 | 0.34 | 0.64 | 0 | 0.16 | 0.24 | 0.03 | 0.14 | 0.15 | 0.09 | 0.38 | 0.54 |
| search | 0.08 | 0.46 | 0.53 | 0.28 | 0.56 | 0.74 | 0.45 | 1 | 1 | 0 | 0.34 | 0.8 | 0 | 0 | 0.05 | 0.02 | 0.18 | 0.24 | 0.11 | 0.52 | 0.69 |
| GOBNILP | 0.51 | 0.75 | 0.81 | | OT | | 0.7 | 1 | 1 | 0.2 | 0.46 | 1 | 0.07 | 0.09 | 0.36 | 0.24 | 0.44 | 0.66 | | OT | |
| cmi-d | 0.69 | 0.85 | 0.86 | 0.64 | 0.91 | 0.99 | 0.75 | 1 | 1 | 0.58 | 0.7 | 0.86 | 0.58 | 0.72 | 0.93 | 0.29 | 0.63 | 0.67 | 0.6 | 0.85 | 0.86 |
| bf_k2-d | 0.61 | 0.74 | 0.75 | 0.44 | 0.67 | 0.8 | 0.95 | 1 | 1 | 0.46 | 0.62 | 0.58 | 0.32 | 0.2 | 0.22 | 0.28 | 0.28 | 0.3 | 0.82 | 0.84 | 0.85 |

To further analyze the quality of the selected V-structures, we picked "cmi-d" and "bf_k2-d" as representatives and investigated their precision and recall against the baseline methods. In Table 4.4, our two methods generally give better precision compared to the greedy method while the search method achieves very good results. And GOBNILP gives the best results on small networks. Among our methods, "bf_k2-d" produces slightly weaker results, compared to "cmi-d". However, in Table 4.5, our methods, especially "cmi-d", produce much better recall, and even outperform GOBNILP for some datasets. This indicates that our methods can more accurately detect the V-structures in the true network. The key advantage of our method is using the relationship among V-structures encoded in the constraints to help the V-structures selection rather than testing each V-structure alone. On the other hand, the search method finds many true V-structures but it also misses some V-structures as the search method does not consider V-structures specifically.

**On the Learned Skeletons.**

In practice, the true skeleton is not always available so we have to use the learned skeleton instead. The learned skeleton may differ from the true skeleton and contains false positive and false negative errors. Furthermore, the learned skeleton

can be too sparse to give a reasonable number of candidate V-structures, especially when data is limited. Since we cannot select the V-structures whose arcs are missing in the skeleton. We should try to avoid false negative errors. In contrast, false positive errors produce extra edges in the skeleton and allow more V-structure candidates but they do not limit the V-structures selection. This coincides with the so-called super-structures [54] that are learned to cover the true skeleton. The aim of super-structures is to reduce the false negative errors as much as possible at the cost of more false positive errors. By increasing the significance level of the independence tests, the constraint-based algorithms, such as MMPC can be used to produce these super-structures [54]. A specific algorithm for super-structures, Hybrid Parents and Children (HPC) is developed in [55].

TABLE 4.6: Performance in averaged SHD on the skeleton learned from HPC with $10^4$ samples. "OT" means "out of time". "*" means ASOBS is used instead of GOBNILP due to the size of the network.

| | ALAR | ANDE | ASIA | CHIL | MILD | INSU | WIN9 |
|---|---|---|---|---|---|---|---|
| greedy | 12.5 | 134.4 | 2 | 12.2 | 41.7 | 30.8 | 52.5 |
| search | 13.2 | 78.3 | 0.8 | 11.2 | 43 | 41.2 | 52.9 |
| pc | 15.6 | 150.5 | 4.2 | 9.3 | 38 | 33.6 | 58.6 |
| cmi-d | 4.1 | 128.6 | 2.1 | 13.4 | 37 | 29.3 | 47.2 |
| bf_k2-d | 5.2 | 134.4 | 2.1 | 13 | 42.1 | 29.3 | 48.2 |
| GOBNILP/ASOBS | 3.2 | 347.5* | 1.4 | 0.0 | 22.0 | 12.8 | 113.6* |

TABLE 4.7: Performance in averaged precision on the skeleton learned from HPC with $10^4$ samples. "OT" means "out of time". "*" means ASOBS is used instead of GOBNILP due to the size of the network.

| | ALAR | ANDE | ASIA | CHIL | MILD | INSU | WIN9 |
|---|---|---|---|---|---|---|---|
| greedy | 0.89 | 0.89 | 0.88 | 0.69 | 1 | 0.76 | 0.79 |
| search | 0.97 | 1 | 1 | 0.7 | 1 | 0.49 | 0.89 |
| pc | 0.91 | 0.51 | 0.75 | 0.84 | 1 | 0.51 | 0.8 |
| cmi-d | 0.98 | 0.7 | 0.75 | 0.53 | 0.98 | 0.55 | 0.89 |
| bf_k2-d | 0.92 | 0.69 | 0.75 | 0.69 | 1 | 0.65 | 0.88 |
| GOBNILP/ASOBS | 0.99 | 0.45* | 0.92 | 1.00 | 1.00 | 0.83 | 0.57* |

We compared the learned skeleton from IAMB, MMPC and HPC with significance level $\alpha = 0.05$ and 0.5. HPC with the default $\alpha = 0.05$ gives the most accurate skeleton. We then compare our weighting methods on this skeleton obtained from HPC in Table 4.6. We also ran GOBNILP in default settings (no node having more than 3 parents and BDeu score) and the PC algorithm [15] as other baselines. Our methods still beat the greedy method but the improvement is limited due to the imperfect skeleton. Similar to the true skeleton, we keep the advantages

TABLE 4.8: Performance in averaged recall on the skeleton learned from HPC with $10^4$ samples. "OT" means "out of time". "*" means ASOBS is used instead of GOBNILP due to the size of the network.

| | ALAR | ANDE | ASIA | CHIL | MILD | INSU | WIN9 |
|---|---|---|---|---|---|---|---|
| greedy | 0.53 | 0.59 | 0.95 | 0.5 | 0.07 | 0.31 | 0.42 |
| search | 0.53 | 0.63 | 1 | 0.82 | 0.03 | 0.21 | 0.39 |
| pc | 0.48 | 0.38 | 0.45 | 0.7 | 0.11 | 0.34 | 0.26 |
| cmi-d | 0.71 | 0.62 | 0.95 | 0.7 | 0.14 | 0.36 | 0.48 |
| bf_k2-d | 0.67 | 0.59 | 0.95 | 0.52 | 0.07 | 0.3 | 0.45 |
| GOBNILP/ASOBS | 0.81 | 0.32* | 0.95 | 1.00 | 0.32 | 0.66 | 0.37* |

in recall with comparable precision, as shown in Table 4.7 and 4.8. The search method gives a good result because the super-structure from HPC is also good for the search as shown in [56]. For the dataset ANDES and WIN95PTS that GOBNILP cannot handle, we further ran the acyclic selection ordering-based search (ASOBS) [57] once instead, which is an approximate local-search-based method using the R package **r.blip** with settings: time=3600(1 hour), scorefunction ="k2", cores=4. And the PC algorithm gives relatively weak results for most datasets.

## 4.5 Summary

The edge orientation is the next stage of BN structure learning after learning the skeleton. In this chapter, we first proposed two weighting methods that measure the strength of V-structures in terms of the conditional dependence and additional dependence in each V-structure and then formulate V-structures selection in BNs as a weighted MAX-SAT problem. Furthermore, we demonstrate that our approach can jointly test all possible V-structures in the skeleton empirically. This takes the advantages from the competition of the weights for the conditional dependence and additional dependence of each V-structure, through the computed weights and the constraints in the skeleton. The selection of a particular V-structure not only uses the data in this V-structure but also considers data in other V-structures.

# Chapter 5

# Structural Knowledge Transfer for Learning Sum-Product Networks

In chapter 3 and 4, we introduced methods to improving the structure learning for Bayesian network. There are other methods, which are available to improve the structure learning for Bayesian network. Transfer learning is one of them and a few papers have studied transfer methods on the structure learning for Bayesian network and shown good progress [58–60]. On the other hand, other probabilistic models are continuously being proposed. Sum-Product Network (SPN) is a recently proposed probabilistic graphical model with a deep architecture [21] that has a network structure in its model representation. Moreover, there is a strong connection between Bayesian network and sum-product network. Under certain conditions, a sum-product network can be converted to a Bayesian network [61]. Therefore, we would like to investigate whether transfer learning can help the structure learning for sum-product network as well.

To overcome small training dataset issues in learning inference model, transfer learning [62] can be utilized to improve the performance and efficiency of a learning (target) task with the help of knowledge from other related (source) tasks. Conventional transfer learning settings assume the availability of either labelled or unlabelled data from the source tasks [62]. However, when one is given only a source inference model instead of data instances from a source task, the problem becomes more challenging.

In this chapter, our proposed transfer learning approach TopTrSPN first utilizes the top layer clustering information of the source SPN structure to lay the foundation for the target SPN structure. Then, the deeper layers in the SPN structure are learned using the target data. With transfer learning, we use the target data with the top layer of the source SPN structure because limited target data cannot learn a good top layer of the SPN. In the learning process of LearnSPN, the algorithm actually makes multiple uses of data by clustering and partitioning the big data blocks into small data blocks recursively to form the layer structures of SPN. Different from LearnSPN and Bayesian network structure learning in Chapter 3 and 4 , we do not make multiple uses of the same data at different stages or layers but use the source SPN at the top layer clustering and the target data at the deeper layers learning.

This chapter is from our published paper [63] and organized as follows. In Section 5.1, we present related work on transfer learning for inference models with network structure. In Section 5.2, we define our problem setting and state our assumptions. In Section 5.3, we describe in detail our proposed transfer learning approach for SPN given a source structure and a limited number of target examples. In Section 5.4, we present and discuss empirical results, and further discuss our investigations in deeper layer transfer. Finally, in Section 5.5, we summarize our main results of this chapter.

## 5.1 Related Work

Transfer learning approaches have been proposed for learning network structure, including Bayesian network [60]. In general, the approaches can be categorized into three types. The first one is transfer learning at data instance level. The source data is transformed to be coherent with the target data. The transformed source data is viewed as additional target data, with updated weights [64]. By combining the transformed source data and target data, one obtains a larger dataset. Towards this end, the standard structure learning can be performed more accurately using this larger dataset.

The second way is transferring the knowledge in the structure search space. This can be a score-based approach, which searches for structures with high scores based

on conditional probability [58, 59]. For Bayesian network, in [58], the posterior probability $P(\mathbf{G}|\mathcal{D})$ of structure $\mathbf{G}$ given data $\mathcal{D}$, is used as a score to enable heuristic search for a better Bayesian network structure. $P(\mathbf{G}|\mathcal{D})$ is obtained from the prior distribution of structures $P(\mathbf{G})$ and data. Transfer learning is attained through $P(\mathbf{G})$, defined to encourage similar structures for related tasks. Shapiro et. al [65] suggested using transferred network structures to reduce the number of structures to be searched for a target task. Mihalkova and Mooney [66] also followed this scheme to perform transfer learning on Markov logic networks.

Finally, transfer learning can be performed at model level. In particular, the source domain knowledge is integrated into the structure learning algorithm. For example, Luis et al. [67] modified the PC algorithm [15] by incorporating knowledge in related tasks. The main idea of the PC algorithm is to perform local independence tests between each pair of variables to determine whether there should be an edge between them in the Bayesian network. A structural similarity measure is used in the independence test as a criterion for structural transfer learning.

## 5.2 Problem Setting and Assumption

A domain is defined by two components: a feature space of data $\mathcal{X}$ and a marginal probability distribution $P(X)$ [62], where $X = \{x_1, \ldots, x_k\} \in \mathcal{X}$ and $k$ is the number of data instances. The source and target domains are represented by $\mathcal{D}_S$ and $\mathcal{D}_T$, respectively.

The dimension of the source and target data can be different. An SPN $\mathcal{S}_S$ is learned from the source training data (not disclosed) and an SPN $\mathcal{S}_T$ is learned from the target training data and auxiliary knowledge from $\mathcal{S}_S$. While discriminative learning is possible for SPNs [68], we only consider the generative SPNs here. We have the following assumption to simplify our transfer learning problem setting:

**Assumption:** *The feature space $\mathcal{X}$, is shared by the source domain $\mathcal{D}_S$ and the target domain $\mathcal{D}_T$ completely or partially, i.e., $\mathcal{X}_S \cap \mathcal{X}_T \neq \emptyset$, but the marginal probability distributions are different, i.e., $P_S(X) \neq P_T(X), X \in \mathcal{X}_S \cap \mathcal{X}_T$. Moreover, the number of instances $k_T$ in $\mathcal{D}_T$ is much smaller than $k_S$ in $\mathcal{D}_S$, i.e., $k_T \ll k_S$.*

In this chapter, our problem setting assumes that only the source SPN structure is available without any source training data. We proposed an approach that transfers SPN structure knowledge in the source domain to improve the performance of an SPN for the target domain with limited training examples. It belongs to the third way of transfer learning described in Section 5.1.

## 5.3 Proposed Transfer Learning Approach

### 5.3.1 Motivation

SPNs can be viewed as "probabilistic, general-purpose convolution networks" [21]. Earlier work on transfer learning on convolution neural networks (CNN) [69] shows that (i) first layer features learned from CNN are general for the data domain, and (ii) transferability is negatively affected by "the specialization of higher [or deeper] layer features to the original [source] task". The main difference between CNNs and SPNs is their representations: the former uses features and the later uses clusters. Based on the first point, the source data distribution and its representation in SPN through clusters are general information that supports the SPN structure learning for the target task. For an SPN, the distributions become more specific and local in the lower layers of the network structure. As a result, irrelevant local information from the source domain may negatively impact the SPN construction for the target domain. Based on the above two characteristics, our proposed SPN transfer learning approach transfers the most general distribution information in the form of top layer clusters in the source SPN. To obtain the top layer clusters in the target SPN, we utilize the target dataset and the centroids of the clusters in the source SPN.

Another reason why the top layer clustering has the most significant effect on SPN is that once the instances in the top layer are clustered into a wrong cluster it cannot be recovered in the lower layers. Hence, it weakens the SPN performance. Suppose we have four binary variables $X_1, \ldots, X_4$, $X_i \in \{0, 1\}$ assigned to one of two clusters, $c_1$ and $c_2$. The probability for an instance to be in each cluster is the same, $P(C = c_1) = P(C = c_2) = 0.5$. For $c_1$, let $P(X_1 = 1) = P(X_2 = 1) = P(X_3 = 1) = 0.9$ and $P(X_4 = 1) = 0.01$; for $c_2$, let $P(X_1 = 1) = 0.2$ and $P(X_2 = 1) = P(X_3 = 1) = P(X_4 = 1) = 0.9$. For simplicity, the four variables are

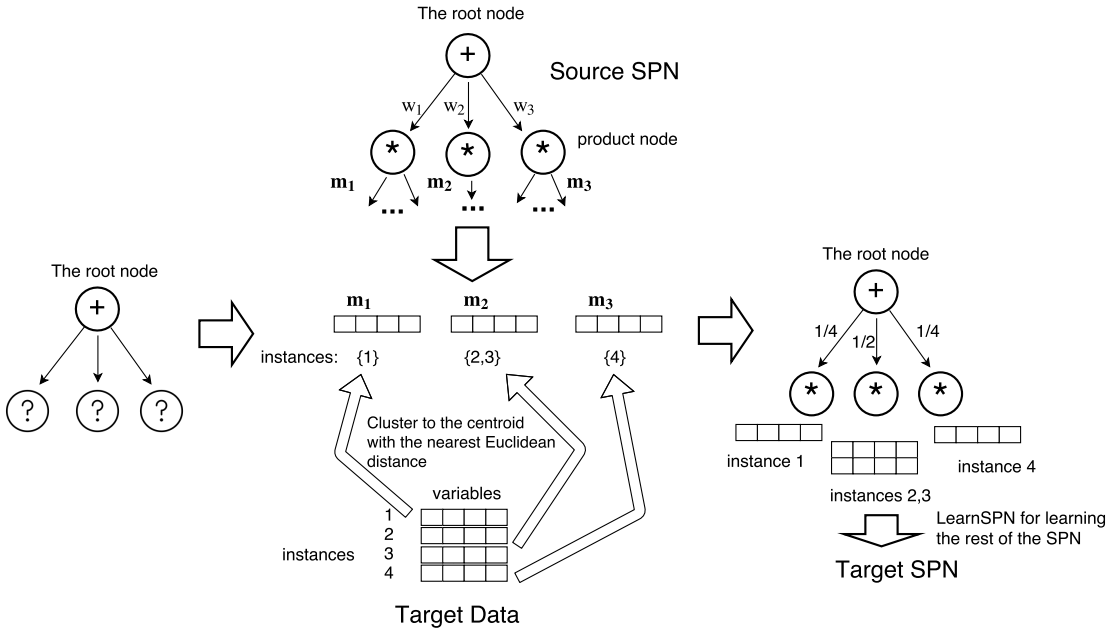



FIGURE 5.1: An example illustrating how TopTrSPN learns the first layer of the SPN for data instances {1, 2, 3, 4} with the help of the centroids $m_1, m_2$ and $m_3$ generated from the source SPN.

assumed to be independent. For a given instance $\hat{\mathbf{x}} = (x_1, x_2, x_3, x_4) = (1, 1, 1, 1)$, it is more likely to be in $c_2$, since $0.9^3 \cdot 0.2 > 0.9^3 \cdot 0.01$, but we do not have this information when we first perform clustering to construct an SPN. We may wrongly cluster $\hat{x}$ with instances coming from $c_1$ but are close to it, such as $\bar{\mathbf{x}} = (1, 1, 1, 0)$, especially when we have only a few training instances. This will lead to an inaccurate representation of the distribution. While we may learn lower layers in the SPN for a subset of variables, it cannot recover the mistake of the first clustering for the top layer as the instance $\hat{x}$ will still be with instances from $c_1$. On the other hand, if we have the centroids generated from our procedure, $(0.9, 0.9, 0.9, 0.01)$ and $(0.2, 0.9, 0.9, 0.9)$ then clustering is simplified and more accurate. In practice, we only have finite samples and the source SPN to approximate these centroids to improve the clustering performance.

An illustration of how our proposed TopTrSPN approach works is shown in Figure 5.1. To learn the SPN structure for the target data instances {1, 2, 3, 4}, we first compute the centroids $\mathbf{m_i}$, ($i = 1, 2$, and 3) from the samples generated from the source SPN. Then we learn the first layer by clustering the instances {1, 2, 3, 4} to their nearest centroids. The weights are based on the number of instances in each cluster. The lower layers of the SPN is learned from the clustered target

instances using LearnSPN. We describe in detail the proposed TopTrSPN approach in Section 5.2.

There are many variants of SPN implementations such as ID-SPN [70] and LearnSPN variant [22], we use the vanilla version of LearnSPN [71] in our algorithm and implementation to demonstrate the feasibility of our proposed transfer approach.

### 5.3.2 LearnSPN and Transfer learning algorithm

**Algorithm 6: LearnSPN($\mathbf{T}, \mathbf{V}$)**

**Input:** set of instances $\mathbf{T}$, set of variables $\mathbf{V}$
**Output:** an SPN $\mathcal{S}$ representing a distribution over $\mathbf{V}$ learned from $\mathbf{T}$.

1: **if** $|\mathbf{V}| = 1$ **then**
2: **return** univariate distribution estimated from the variable's values in $\mathbf{T}$
3: **else**
4: partition $\mathbf{V}$ into approximately independent subsets $\mathbf{V}_j$
5: **if** success **then**
6: **return** $\prod_j$ LearnSPN($\mathbf{T}$, $\mathbf{V}_j$)
7: **else**
8: partition $\mathbf{T}$ into subsets of similar instances $\mathbf{T}_i$
9: **return** $\sum_i \frac{|\mathbf{T}_i|}{|\mathbf{T}|}$ LearnSPN($\mathbf{T}_i, \mathbf{V}$)
10: **end if**
11: **end if**

Before introducing our algorithm, we would like to present the structure learning algorithm, LearnSPN [71], shown in Algorithm 6, which learns the SPN structure by recursively partitioning variables and data instances. The partitioning of variables by independence testing corresponds to a product node (lines 4-6) whereas the partitioning of data instances by clustering (lines 8-9) corresponds to a sum node with weights attached to its children. The partitioning continues and operates on subsets of variables and data instances. The learning ends when there is only one variable in each partition (lines 1-2), which forms a leaf node.

Algorithm 7 shows the steps of TopTrSPN which include the two key steps.

1. **DiffVariable**($\mathbf{T}$, $\mathbf{V}$, $\mathcal{S}_S$): filter out those variables in the source SPN $\mathcal{S}_S$ that have a marginal probability that is different from the target domain when calculating the nearest centroids.

2. **TransCluster**($\mathbf{T}$, $\mathcal{S}_S$, $\mathbf{V}_{ignore}$): partition the set of data instances $\mathbf{T}$ into subsets of similar instances $\mathbf{T}_i$ based on the nearest centroids provided by the clusters in the top layer of $\mathcal{S}_S$.

Note that by removing **DiffVariable**($\mathbf{T}$, $\mathbf{V}$, $\mathcal{S}_S$) and **TransCluster**($\mathbf{T}$, $\mathcal{S}_S$, $\mathbf{V}_{ignore}$), i.e. line 8 - 12, Algorithm 7 becomes LearnSPN, as shown in Algorithm 6.

**TopTrSPN**

**Algorithm 7: TopTrSPN($\mathbf{T}, \mathbf{V}, \mathcal{S}_S$)**

**Input:** set of instances $\mathbf{T}$, set of variables $\mathbf{V}$, the source SPN $\mathcal{S}_S$
**Output:** The target SPN $\mathcal{S}_T$ representing a distribution over $\mathbf{V}$ learned from $\mathbf{T}$.

1: **if** $|\mathbf{V}| = 1$ **then**
2: **return** univariate distribution estimated from the variable's values in $\mathbf{T}$
3: **else**
4: partition $\mathbf{V}$ into approximately independent subsets $\mathbf{V}_j$
5: **if** success **then**
6: **return** $\prod_j$ LearnSPN($\mathbf{T}$, $\mathbf{V}_j$)
7: **else**
8: **if** doing the first clustering **then**
9: $\mathbf{V}_{ignore} \leftarrow$**DiffVariable**($\mathbf{T}$, $\mathbf{V}$, $\mathcal{S}_S$)
10: $\{\mathbf{T}_i^K\}_{i=1} \leftarrow$ **TransCluster**($\mathbf{T}$, $\mathcal{S}_S$, $\mathbf{V}_{ignore}$)
11: **return** $\sum_i \frac{|\mathbf{T}_i|}{|\mathbf{T}|}$ LearnSPN($\mathbf{T}_i, \mathbf{V}$)
12: **else**
13: partition $\mathbf{T}$ into subsets of similar instances $\mathbf{T}_i$
14: **return** $\sum_i \frac{|\mathbf{T}_i|}{|\mathbf{T}|}$ LearnSPN($\mathbf{T}_i, \mathbf{V}$)
15: **end if**
16: **end if**
17: **end if**

As described in Section 5.3.2, LearnSPN [71] consists of two recursive steps, namely: (i) clustering data instances and (ii) partitioning variables based on independence testing. The internal sum nodes and product nodes are obtained by recursively and alternatively doing these two steps. This scheme means that different clustering methods and independence tests can be applied. We follow their settings [71], which are the Expectation-Maximization (EM) algorithm for clustering and the G-test for pairwise independence testing [72], for all layers in an SPN except the top layer.

To perform transfer learning for the target SPN constructed using LearnSPN, an intuitive approach is to utilize clustering and independence test approaches to

match similar variables. This can be done directly on the top layer of the SPN (i.e., the layer directly under the root node). Our proposed approach, TopTrSPN, transfers the source SPN information from the top layer cluster centroids to help the target SPN learning when only limited target data is available.

As one traverses deeper down the source SPN, the correspondence between the source SPN $\mathcal{S}_S$ and the target SPN $\mathcal{S}_T$ starts to weaken. This is because the variables contained in each node in the same layer start to differ significantly due to the large number of possible variable set partitions that do not resemble the source SPN. Further discussion on the weaknesses of transfer learning in the lower layers is deferred to Section 5.4.4.

One notes that the first clustering to get the children of the root node using the EM algorithm takes up a large proportion of the total training time. The expensive first clustering step is due to a large number of instances and variables at the beginning of the learning process. As the structure continues to be built, the number of instances and variables involved reduces, leading to faster clustering and independence testing.

**Algorithm 8: DiffVariable($\mathbf{T}, \mathbf{V}, \mathcal{S}_S$)**

**Input:** set of instances $\mathbf{T}$, set of variables $\mathbf{V}$, the source SPN $\mathcal{S}_S$
**Output:** a set of variables $\mathbf{V}_{ignore}$.

$\mathbf{V}_{ignore} \leftarrow \emptyset$
Compute the marginal probabilities $\mathbf{p_t}$ from $\mathbf{T}$ and $\mathbf{p_s}$ from $\mathcal{S}_S$ for each variable $i$ from $\mathbf{V}$.
$\mathbf{d} \leftarrow \mathbf{p_t} - \mathbf{p_s}$
MAD $\leftarrow$ median$\{|\mathbf{d} - \text{median}(\mathbf{d})|\}$.
**for** each $X_i \in \mathbf{V}$ **do**
    The modified Z-score $M_i \leftarrow (0.6745 * (d_i - \text{median}(\mathbf{d})))/\text{MAD}$ .
    **if** $|M_i| > 3.5$ **then**
        $\mathbf{V}_{ignore} \leftarrow \mathbf{V}_{ignore} \bigcup \{X_i\}$
    **end if**
**end for**
**return** $\mathbf{V}_{ignore}$

### DiffVariable

There may exist a discrepancy between the source domain and the target domain. For example, some variables have different (or shifted) distributions between the

source domain and the target domain. Those variables showing significant differences between the marginal distributions in the two domains are filtered out in the centroid estimation to ensure that the clusters for the target SPN are not skewed by these variables. To identify such variables, the marginal distributions from the source structure and the target data are compared for each variable. To find a significant difference between the two marginal distributions, we identify the outliers among those differences.

Let $\mathbf{p_t}$ and $\mathbf{p_s}$ be the vectors of the marginal probability $[P(X_1 = 1), P(X_2 = 1), \ldots, P(X_n = 1)]$ for the target data and the source SPN, respectively. Let

$$\mathbf{d} = \mathbf{p_t} - \mathbf{p_s} \tag{5.1}$$

be the difference between $\mathbf{p_t}$ and $\mathbf{p_s}$.

For any variable $X_i$ whose $d_i$ is in the region of outliers, its values in the centroids of the top layer nodes are ignored and simply set to zero so that it does not affect the clustering process.

We use the modified Z-Score $M_i$ for variable $X_i$, computed from $\mathbf{d}$ and the criteria $|M_i| > 3.5$ to detect the significant difference [73]. The advantage of the modified Z-Score is that it avoids the effects of a few extreme values on the estimated mean and variance by using the median and the median of the absolute deviation (MAD) of the median instead.

For the case that two different domains have different numbers of features where only some of the features are shared between these two domains, we perform a direct matching for these shared features. The matched features are used when computing the distance between the instance and the centroid in the top layer clustering process.

### TransCluster

TransCluster in algorithm TopTrSPN transfers only the top layer clustering information in the source SPN and TopTrSPN learns the rest of the structure using the target data and the original LearnSPN with the assumption that the source domain is similar to the target domain of interest. Our algorithm clusters the target instances to their nearest centroids obtained from the top layer nodes in

**Algorithm 9: TransCluster($\mathbf{T}, \mathcal{S}_S, \mathbf{V}_{ignore}$)**

**Input:** set of instances $\mathbf{T}$, the source SPN $\mathcal{S}_S$ and a set of variables $\mathbf{V}_{ignore}$
**Output:** a collection of sets of instances $\{\mathbf{T}_i\}_{i=1}^{K}$.

```
Use S_S to generate samples and compute centroid m_i for cluster i in the top layer of S_S.
K ← Number of top layer clusters in S_S.
for each i = 1, ..., K do
    T_i ← ∅
end for
for each X_j ∈ V_ignore do
    for each i = 1, ..., K do
        m_i[j] ← 0
    end for
end for
for each t ∈ T do
    î ← argmin_i EucildeanDistance(t, m_i)
    T_î ← T_î ∪ {t}
end for
return {T_i}_{i=1}^K
```

the source SPN, where the Euclidean distance measure is used. The centroids are the generated sample means of the corresponding nodes. Note that although samples are generated from the source SPN, they are not directly used in the target SPN learning. The sample size can vary since they are only used to compute the centroids.

Figure 5.1 shows an example where four data instances from the target domain are used to learn the first layer of the target SPN based on the source SPN. Three clusters are learned from the source SPN and the target data. The number of instances in each cluster determines the corresponding weight in the learned SPN.

### 5.3.3 Complexity Analysis

We briefly analyze the computational complexity of **DiffVariable** and **TransCluster**. Suppose the number of instances is $k$, the number of variables is $n$, the number of clusters in the top layer of the source SPN $\mathcal{S}_S$ is $K$ and the number of nodes in $\mathcal{S}_S$ is $m(m > n)$.

The running time of **DiffVariable** is dominated by the querying of the source SPN $\mathcal{S}_S$ for the marginal probabilities. The running time of a single query grows linearly

as the size of $\mathcal{S}_S$, $m$ increases due to the property of SPN [21]. Hence, the total running time is $O(nm)$.

For **TransCluster**, the number of samples generated to compute the centroids is flexible. The running time of setting zeros is at most $O(Kn)$. A significant amount of the running time is used to find the nearest centroid cluster for each instance. The running time for one instance is $O(Kn)$. The total running time is $O(Knk)$.

## 5.4 Experimental Results

In this section, we present experimental results comparing our algorithms with some existing SPN implementations on the twenty binary datasets used in [71]. We also present results on the knowledge transfer between two real-world text datasets, 20 Newsgroups and Reuters-52, where there is a difference in their marginal distributions. Most importantly, the set of variables from the two text datasets are not the same. In Section 5.4.1, we briefly describe the datasets used. In Section 5.4.2, we describe the experimental settings for the approaches we compared. In Section 5.4.3, we discuss in detail our empirical results. In Section 5.4.4, we discuss our investigations on lower layer transfer for SPN.

### 5.4.1 Datasets.

Among the twenty binary datasets, the number of data instances ranges from 2,225 to 388,434 and the number of variables ranges from 16 to 1,556. For most datasets, 10% of instances are used for validation and 15% are used for testing [71]. Each instance is a binary vector. The binary value indicates whether a certain feature exists or not in the instance.

To investigate transfer learning between two tasks with a different set of variables, we use the text datasets 20 Newsgroups and Reuters-52 which correspond to C20NG and CR52, respectively, in the binary datasets. We extracted the binary features that represent whether a certain word appears in a document or not, under the same criteria as in [74], i.e., for 20 Newsgroups, only words that appeared in at least 200 documents are taken as a feature. For Reuters-52, words are required to appear in at least 50 documents to be a feature. We obtained a slightly different

TABLE 5.1: Average test log-likelihoods for 20 binary datasets, 20 Newsgroups and Reuters-52.

| Dataset | LearnSPN | ID-SPN-t | SPN-t | TopTrSPN | Top2TrSPN |
|---|---|---|---|---|---|
| EachMovie | -52.485 | -59.760 | **-58.727** | -59.743 | -60.325 |
| MSWeb | -10.252 | -10.685 | -11.040 | **-10.876** | -11.036 |
| KDD | -2.182 | -2.199 | **-2.246** | -2.269 | -2.231 |
| Audio | 40.503 | -41.652 | -42.112 | **-41.451** | -41.427 |
| Book | -35.886 | -37.974 | -36.927 | **-36.844** | -36.780 |
| Jester | -75.989 (-53.405) | -55.006 | -55.211 | **-54.396** | -54.391 |
| MSNBC | -6.113 | -6.093 | -6.155 | **-6.133** | -6.211 |
| Netflix | -57.328 | -58.465 | -59.075 | **-58.401** | -58.428 |
| NLTCS | -6.110 | -6.159 | **-6.203** | -6.210 | -6.211 |
| Plants | -12.977 | -13.531 | **-14.754** | -14.810 | -15.092 |
| Accidents | -30.038 | -30.471 | -32.248 | -32.329 | -35.447 |
| Ad | -19.733 | -58.030 | -47.460 | **-45.444** | -52.160 |
| BBC | -250.687 | -282.126 | -276.081 | **-275.554** | -276.213 |
| C20NG | -155.925 | -161.526 | -162.617 | **-160.364** | -160.520 |
| CWebKB | -158.204 | -183.524 | -172.477 | **-169.286** | -170.404 |
| DNA | -82.523 | -100.990 | -87.514 | -88.622 | -93.728 |
| Kosarak | -10.989 | -11.199 | **-11.195** | -11.654 | -11.610 |
| Retail | -11.043 | -11.393 | -11.416 | **-11.314** | -11.324 |
| Pumsb-Star | -24.781 | -26.052 | **-27.291** | -27.593 | -31.929 |
| CR52 | -85.067 | -98.099 | -95.335 | -95.265 | -99.194 |
| 20 Newsgroups | -158.782 | -161.881 | -163.748 | **-162.782** | -162.733 |
| Reuters-52 | -87.168 | -99.607 | -95.907 | **-94.629** | -94.627 |

number of features (in brackets) from 20 Newsgroups (916) and Reuters-52 (908) compared to C20NG (910) and CR52 (889). We keep the partitions of training, validation, and testing unchanged. For 20 Newsgroups and C20NG, the partitions are (11293, 3764, 3764); for Reuters-52 and CR52, the partitions are (6532, 1028, 1540).

### 5.4.2 Experimental Settings.

1. **LearnSPN** [71]: The full training data is used to obtain the results. The results shown in Table 5.1 are from [71]. [1]

2. **ID-SPN-t** : ID-SPN [70] is used on 10% of the training data as the target domain without any source structure.

[1]For the dataset Jester, there is a big difference between the result reported in [71] and our result in the bracket reusing their codes.

3. **SPN-t**: LearnSPN is used on 10% of the training data as the target domain without any source structure.

4. **TopTrSPN**: The target data is the same 10% training data as in SPN-t. Source SPNs are learned by LearnSPN with the other 90% of the training data, i.e. no overlapping with the training data in SPN-t, and 20% of variables are replaced by Bernoulli random variables with a uniform random $p \in [0, 1]$, introducing variations in the marginal distributions.

5. **Top2TrSPN**: Top2TrSPN (see Section 5.4.4) is similar to TopTrSPN, but it has a second layer transfer, i.e., the first layer of independence partitioning, copied from the source SPN.

When we applied TopTrSPN to 20 Newsgroups and Reuters-52, they took each other's SPN learned with full training data using LearnSPN as the source SPN. We matched the words in the two datasets and the same words appearing in both datasets became shared features across the two domains. During the top layer clustering transfer, we only consider those shared words. For Top2TrSPN, we use those independence partitions of shared words in the source SPN as the seeds for the target independence partitioning. The experiments were performed on Intel Xeon E5-1650 Processor(Six Core HT, 3.2GHz Turbo) with 16GB of RAM.

### 5.4.3 Results and Discussions.

We follow prior work on SPN structure learning [71] and take the test-set log-likelihood and query conditional log-likelihood as measures of SPN performance. Furthermore, we add the training time to compare the learning efficiency.

For ID-SPN-t and SPN-t, we repeated 10 trials and took the average. For TopTrSPN and Top2TrSPN, we use 5 different source SPNs and repeated 10 times for each source SPN. We take SPN-t as our baseline model and compare TopTrSPN with it. Significant results in TopTrSPN are identified with a two-sample t-test against SPN-t with $p = 0.05$ and shown in bold in Table 5.1 and 5.2.

#### Test-set Log Likelihood

Table 5.1 [2] shows the average test-set log-likelihood of LearnSPN, SPN-t, TopTrSPN, and Top2TrSPN. We present results for all datasets in [71] without cherry-picking in favor of our proposed approaches. Here, LearnSPN serves as an upper bound of the performance. In eleven out of twenty datasets, we obtain a significantly better SPN by utilizing TopTrSPN on the source SPN than SPN-t.

Some additional observations from Table 5.1 are as follows:

- There are three datasets (Accidents, DNA, CR52) which our approaches do not improve the performance of SPN-t. This is due to the lack of useful structural information being transferred from the source SPN structure in the first layer clustering step.
- There are six datasets (EachMovie, KDD, NLTCS, Plants, Kosarak, Pumsb-Star) that have worse results than SPN-t when both our approaches are applied. This is due to the negative transfer [62] from the source SPN which could be due to our experimental setting when up to 20% of the variables have their marginal distribution changed.
- The test-set log-likelihoods of using LearnSPN (based on the original training datasets) for all the datasets are significantly better than performing transfer learning from a source SPN. This should not be surprising as the source SPN does not exactly model the dataset due to our 20% variations of the variables.

For 20 Newsgroups and Reuters-52, TopTrSPN shows significantly better results. We further investigate the effect of the target data size on 20 Newsgroups shown in Figure 5.2 and Reuters-52 shown in Figure 5.3 with error bars for SPN-t and TopTrSPN. In both Figure 5.2 and Figure 5.3, for both methods, the log-likelihood tends to increase as the size of target data increases from 10% to 50%. However, in Figure 5.2, there is a drop in performance when the target data size is 30%. This may be caused by the conflict between the source SPN and the target data. When either the source SPN or the target data dominates, the performance improves. When their strengths are even, it may lead to a worse model.

When the target data is very limited, e.g. 10%, our methods with the help from the source SPN perform better than SPN-t. However, as the size of the target data

[2] The result for Jester dataset reported in [71] for LearnSPN is -75.989. It is likely to be a typo. We reported it as -53.405.

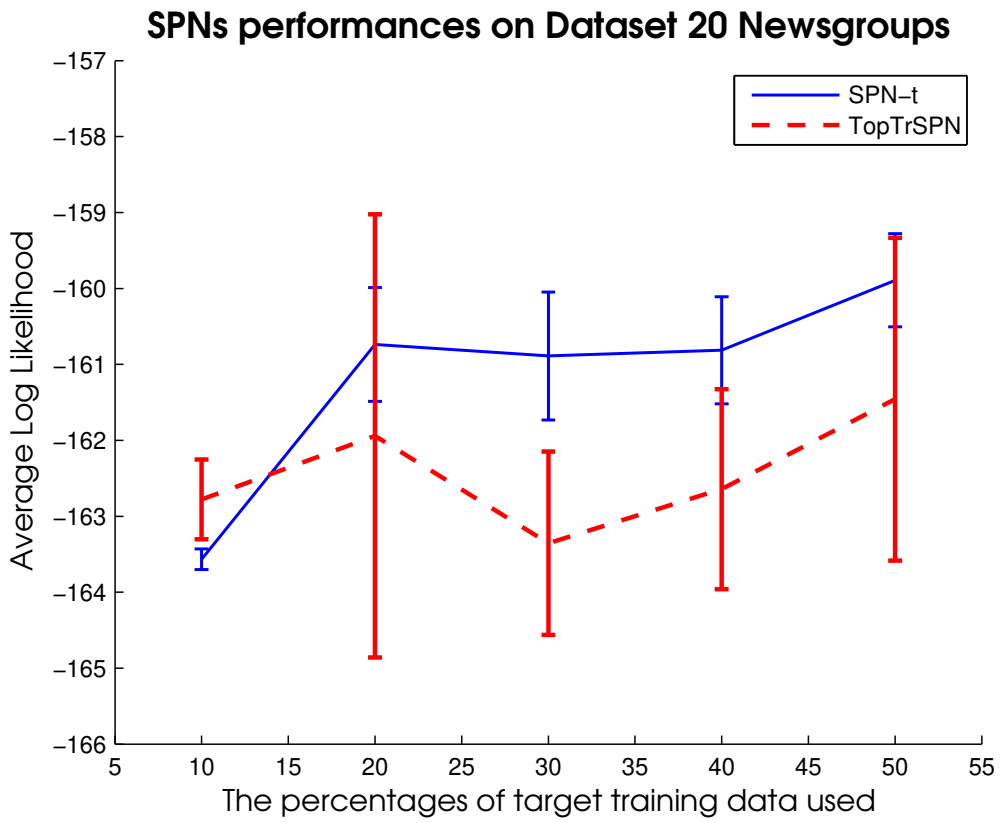



FIGURE 5.2: Average test log-likelihoods for 20 Newsgroups given Reuters-52 as source SPN.

rises our methods start to lose their advantage over SPN-t which directly learns from the target data. In particular, in Figure 5.3, the standard deviation of SPN-t decreases while the one for TopTrSPN increases as data size grows. This is because more target data makes the source SPN information less valuable. If enough target data is available, then the source SPN may not help but mislead the target SPN and reduce its performance.

### Query Conditional Log Likelihood

Queries are generated with different proportions of query variables $\mathbf{Q}$ and evidence variables $\mathbf{E}$. A set of instances are randomly selected from the test set in each dataset. Then, queries $P(\mathbf{Q} = \mathbf{q}|\mathbf{E} = \mathbf{e})$ are created by randomly fixing proportions of query variables $\mathbf{Q}$ and evidence variables $\mathbf{E}$. These results are used to compare the inference performance of the SPNs. Results are normalized by the number of query variables.

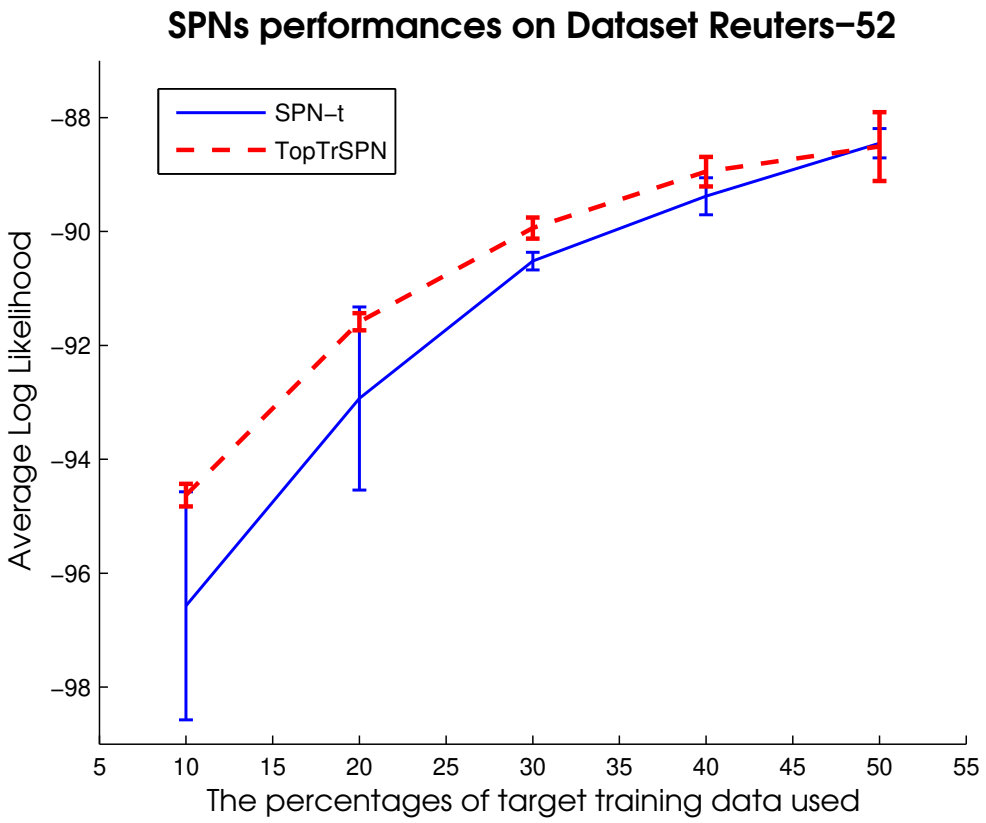



FIGURE 5.3: Average test log-likelihoods for Reuters-52 given 20 Newsgroups as source SPN.

In our experiments, the proportions of query/evidence are fixed as 30%/50% and 50%/30%. The significant results are six and eight out of twenty for TopTrSPN over SPN-t, shown in Table 5.2. Seven datasets have significantly worse results than SPN-t in both query/evidence settings when our approaches are applied. In summary, TopTrSPN does not perform better in query conditional probability. However, the results for TopTrSPN in the two text datasets are better than SPN-t. Note that there are strong correlations between the test-set log-likelihood in Table 5.1 and the query conditional log-likelihoods.

### Training Time

From Table 5.3, TopTrSPN needs less time to train than SPN-t for twelve datasets. This does not sound significant. However, the training time of TopTrSPN includes some pre-computation time of querying the marginal distribution for each variable, sampling 10,000 samples from the source SPN and calculating the centroid. The number in the bracket is the remaining time for learning the SPN structure. If we

TABLE 5.2: Average conditional log-likelihoods for 20 binary datasets, 20 Newsgroups, and Reuters-52. Results are normalized by the number of query variables.

| Q./Ev. % | 30%/50% | | 50%/30% | |
|---|---|---|---|---|
| Dataset | SPN-t | TopTrSPN | SPN-t | TopTrSPN |
| EachMovie | **-0.1135** | -0.1160 | **-0.1149** | -0.1170 |
| MSWeb | **-0.0335** | -0.0337 | **-0.0326** | -0.0327 |
| KDD | -0.0287 | -0.0286 | -0.0304 | -0.0305 |
| Audio | -0.4041 | **-0.3969** | -0.4109 | **-0.4032** |
| Book | **-0.0696** | -0.0702 | **-0.0689** | -0.0695 |
| Jester | -0.5395 | **-0.5310** | -0.5389 | **-0.5309** |
| MSNBC | -0.3503 | -0.3499 | -0.3148 | **-0.3145** |
| Netflix | -0.5815 | **-0.5743** | -0.5842 | **-0.5770** |
| NLTCS | **-0.2693** | -0.2710 | -0.3519 | -0.3522 |
| Plants | **-0.1730** | -0.1751 | **-0.1818** | -0.1825 |
| Accidents | -0.2464 | -0.2471 | -0.2616 | -0.2625 |
| Ad | -0.0263 | **-0.0258** | -0.0272 | **-0.0262** |
| BBC | **-0.2598** | -0.2605 | -0.2585 | **-0.2583** |
| C20NG | -0.1812 | **-0.1787** | -0.1816 | **-0.1791** |
| CWebKB | -0.2014 | **-0.1990** | -0.2009 | **-0.1977** |
| DNA | -0.4558 | -0.4659 | -0.4729 | -0.4798 |
| Kosarak | **-0.0558** | -0.0571 | **-0.0574** | -0.0587 |
| Retail | **-0.0829** | -0.0840 | **-0.0807** | -0.0817 |
| Pumsb-Star | **-0.1256** | -0.1281 | **-0.1397** | -0.1419 |
| CR52 | -0.1027 | -0.1026 | -0.1029 | -0.1027 |
| 20 Newsgroups | -0.1779 | **-0.1771** | -0.1795 | -0.1794 |
| Reuters-52 | -0.1055 | **-0.1044** | -0.1033 | **-0.1020** |

only consider and compare these remaining learning time, then seventeen datasets are learned faster.

For all twenty datasets except Ad, Top2TrSPN has lower training time compared with TopTrSPN. There is a big training time saving if we directly copy the source independence partitions in the first independence partitioning layer. The tradeoff for this saving is a possible degradation in performance (see Table 5.1).

### 5.4.4 Limitations in Transferring Lower Layer Information

While our proposed method utilizes the clustering information in the top layer of the source SPN, we also investigate transferring information from the lower layers in the SPN. We implement a variant of TopTrSPN, called Top2TrSPN, which after

TABLE 5.3: Average training time (seconds) for 20 binary datasets, 20 Newsgroups and Reuters-52. The numbers in the brackets are the time of only learning the SPN structure without precomputations.

| Dataset | SPN-t | TopTrSPN | Top2TrSPN |
|---|---|---|---|
| EachMovie | 6.2 | 7.5 (4.9) | 4.9 |
| MSWeb | 10.4 | 8.2 (6.9) | 3.7 |
| KDD | 15.5 | 8.6 (3.8) | 5.9 |
| Audio | 32.1 | 6.4 (5.8) | 1.0 |
| Book | 6.6 | 6.6 (4.8) | 2.8 |
| Jester | 32.0 | 9.2 (8.5) | 1.0 |
| MSNBC | 96.7 | 42.8 (32.7) | 12.6 |
| Netflix | 48.4 | 16.5 (15.5) | 1.9 |
| NLTCS | 1.8 | 0.9 (0.8) | 0.2 |
| Plants | 7.7 | 4.8 (4.4) | 1.1 |
| Accidents | 55.2 | 6.1 (5.2) | 3.7 |
| Ad | 120.1 | 135.9 (10.9) | 184.1 |
| BBC | 8.5 | 31.0 (6.0) | 27.2 |
| C20NG | 54.5 | 204.3 (42.6) | 173.7 |
| CWebKB | 8.9 | 24.4 (8.6) | 18.1 |
| DNA | 2.9 | 1.4 (1.1) | 0.8 |
| Kosarak | 6.7 | 6.0 (5.0) | 2.0 |
| Retail | 3.2 | 2.4 (1.9) | 0.9 |
| Pumsb-Star | 10.0 | 10.2 (9.2) | 5.1 |
| CR52 | 36.1 | 71.4 (19.2) | 58.1 |
| 20 Newsgroups | 55.9 | 110.9 (45.4) | 117.0 |
| Reuters-52 | 38.2 | 227.1 (20.7) | 243.5 |

the first clustering in TopTrSPN, copies the first layer of independence sets in the source SPN to the corresponding clusters.

In LearnSPN, the sum nodes of SPNs are learned by clustering instances while the product nodes are learned by partitioning independence sets. To transfer the deeper clustering information, one needs to transfer the partitioning information first. However, there is a difference between these two kinds of information. For clustering information, such as the cluster centroids, the outliers usually do not degrade the clustering of the target data because the cluster centroids from the source SPN are only possible candidates. Unlike the clustering information, the partitioning information among variables is hard to decide between the source SPN and the target data. If we try to deviate from the source partitioning, the independence sets will contain different variables and cannot be matched with the partitioning of the source SPN completely. This affects learning in the lower layers

of the SPN. As a result, the clustering outcomes have different variables from the source centroids.

We attempted two approaches to overcome this issue, but neither of them provides noticeable improvements in the SPN performance compared with TopTrSPN. The first approach extends Top2TrSPN and goes deeper down the network. Top2TrSPN, as mentioned in Section 5.3, copies the independence sets partitions under the top clustering layer from the source SPN to the target SPN. This is similar to the "frozen" step in [69]. This gives some savings in training time mentioned in Section 5.4.3. Furthermore, Top2TrSPN provides us a bunch of sum nodes in the second clustering layer. This allows us to run our method on those sum nodes, which is similar to TopTrSPN on the root node. For the second approach, we exhaustively search all possible centroids in the second layer by combining mean values from different partitions, and clustering instances. We abandon those centroids with few instances assigned and then redo clustering on those instances. Both of them show striking similarity with the observations by [69] that (i) first layer features learned in convolution neural networks are general for the data domain, and (ii) transferability is negatively affected by "the specialization of higher [or deeper] layer features to the original [source] task".

## 5.5 Summary

In this chapter, we investigate how transfer learning can be used to improve an SPN performance when the number of training examples is limited. In particular, we consider a structural transfer learning setting where we do not have a source dataset but a source SPN. We propose a transfer learning approach called TopTrSPN which utilizes the information of the first layer clusters in the source SPN to learn the first layer of the target SPN. The source structure and target data are used at different layers of the SPN structure learning respectively. Furthermore, our algorithms consider the difference between the source SPN and the target domain by further removing variables that have big marginal distribution variations. Empirical results on benchmark datasets show the feasibility of our proposed transfer learning approach for SPN structure learning.

# Chapter 6

# Conclusions and Future Work

In this chapter, we first summarize the contributions of this dissertation and then point out three promising directions for future work.

## 6.1 Summary of Contributions

Probabilistic models are useful models that can handle uncertainty. These models usually require a certain network structure for their model representation. However, structure learning is generally hard. Therefore, in this dissertation, we aimed to develop data-driven methods involving multiple uses of data to improve the structure learning of Bayesian network and sum-product network, as two representatives of the probabilistic models.

In Chapter 3, we investigated the symmetry correction problem in the BN local structure learning algorithms. To make the local structures of each node consistent and obtain the skeleton, we proposed two data-driven symmetry correction methods, which use data to better learn relations between a pair of nodes which suffers from a conflict in their local structures. We not only showed the improvement due to our symmetry correction methods but also investigated the effect of the different combinations of our methods (score-based or constraint-based) and the learning algorithms (score-based or constraint-based).

Chapter 4 moved to the edge orientation, which is the next stage of BN structure learning after obtaining the skeleton. With improved skeleton obtained in Chapter

3, we found that even if the skeleton is learned perfectly, without a good orientation of the edges, the learned DAG can model data badly and stay far away from the true DAG. We considered V-structure as a key concept in the orientation problem since the set of V-structures determines the equivalent class that the DAGs belong to. Given the skeleton is known, we extracted all possible V-structures and let them compete against each other. For each V-structure, this competition helps to decide its existence by utilizing the data from all other V-structures. Then we found that the competition can be formulated as a weighted MAX-SAT problem. The weights should be able to represent the conditional dependence and the dependence in a V-structure. With this in mind, we found two kinds of reasonable weighting methods. The experimental results demonstrated the usefulness of our approach in the edge orientation.

Besides our approaches, one of the other approaches that can improve the structure learning of BN is transfer learning. And SPN has been proposed as a new kind of probabilistic model, which can be converted to BN under certain conditions. So we attempted to improve the structure learning of SPN with transfer learning. LearnSPN is one of the structure learning algorithms for SPN. Based on it, in Chapter 5, we thus proposed a transfer learning approach TopTrSPN that allows one to utilize SPN with limited training examples, given a source SPN. Our proposed TopTrSPN first utilizes the top layer clustering information of the source SPN structure to lay the foundation for the target SPN structure. Then, the deeper layers in the SPN structure are learned using the target data. We can take the learning of the top layer of SPN as one stage and the learning of the deeper layers as another one. With transfer learning, the different sources of data contribute at different stages of the SPN structure learning.

To sum up, in this dissertation, we contributed to novel data-driven methods that improve the structure learning of BN and SPN. We achieved this goal by carefully examining the structure learning algorithms and then replaced the heuristic rule or default procedure with a data-driven method. The experimental results showed that it is promising to extract different aspects of the structural information from data to improve different stages in the structure learning of probabilistic models.

## 6.2 Future Work

There are multiple potential directions for future work. In this section, we present three promising directions and summarize their relations to our contributions in Figure 6.1.

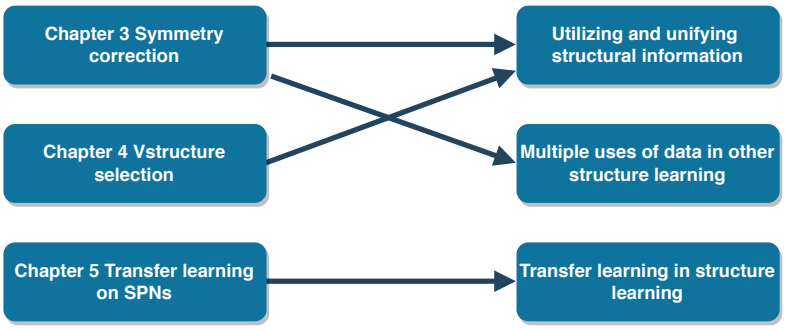



FIGURE 6.1: The relations between future work and our contributions.

**Utilizing and Unifying Structural Information.** One direction is to further utilize data in the later stages of the structure learning and unify them at a higher level. For example, from our investigations on constraint-based and score-based symmetry correction methods for Bayesian network structure learning, the two types of approaches can capture different kinds of structural information. This can be useful in the design of better hybrid learning algorithms and further improving the structure learning. Multiple uses of data in different stages extract different aspects of structural information, which are stored in different intermediate structural results. However, to obtain the final network structure, we should combine and unify these intermediate structural results consistently.

Inspired by [43], besides making multiple uses of data, we can reuse the intermediate structure results in other stages to make the final output network more consistent. In particular, moving backwards from a later stage to improve the early stage is an attractive option. While the later stages naturally depend on the early stage and can benefit from a better early stage result, the early stage is not affected by the later stages. So after the later stage is done, a further backward revision for the early stage is beneficial. For example, the edge orientation results in Bayesian network structure learning can be helpful to refine the skeleton obtained in the early stage. The orientations of edges could suggest adding or removing certain edges in the skeleton. If this refinement improves the skeleton, then we can rerun the later stages and achieve a recursive algorithm, which can run the early stage

and the later stages iteratively. This provides a consistent final network structure, which is a mixture of results from different structural information.

**Multiple Uses of Data in Other Structure Learning.** Another direction is spreading the idea of multiple uses of data at different stages of structure learning to other probabilistic models or more general models. For example, Markov networks, as another kind of probabilistic graphical model [75], is a good candidate. It requires structure learning, so there are opportunities to make multiple uses of data at various stages of an existing structure learning algorithm to improve their performance.

For Markov network, some structure learning algorithm tries to learn the neighborhoods of each variable [76], which is similar to the local structure learning algorithms for Bayesian network. It has to face the symmetry correction problem as well. So the idea of multiple uses of data and our data-driven symmetry correction methods can be extended to this kind of algorithm without much effort.

**Transfer Learning in Structure Learning.** The last direction is designing new transfer learning techniques for structure learning. There have been a few papers on applying transfer learning on other probabilistic models, such as Markov logic network [77], where Markov logic network [78] combines logic reasoning and probability. The aims of applying transfer learning are improving the efficiency to deal with the high computational cost in structure learning and improving the performance of structure learned from limited data. The data or structure in the source domain is what we can use to do the transfer. The structural transfer is harder than the data transfer, but the structural transfer can work across different domains and capture the common structures from very different domains, which may hide under the data. For example, the second-order transfer in Markov logic network discovers new structures in the target domain [79].

The idea of transfer learning is suitable for improving the structure learning, but the traditional transfer learning techniques are not very helpful due to the complexity of structure learning algorithms. So new transfer learning techniques need to be designed for network models and their learning algorithms specifically. This requires a good understanding of the transfer learning, the network model and the corresponding structure learning algorithms.

# List of Publications

## Journal Articles

- **Zhao, Jianjun**, and Shen-Shyang Ho. "Structural knowledge transfer for learning Sum-Product Networks." *Knowledge-Based Systems* 122 (2017): 159-166.
- **Zhao, Jianjun**, and Shen-Shyang Ho. "Improving Bayesian network local structure learning via data-driven symmetry correction methods." *International Journal of Approximate Reasoning* 107 (2019): 101-121.

# Bibliography


[1] Ioannis Tsamardinos, Constantin F Aliferis, and Alexander Statnikov. Time and sample efficient discovery of Markov blankets and direct causal relations. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 673–678. ACM, 2003. 2, 10, 15, 17, 19

[2] Constantin F Aliferis, Alexander Statnikov, Ioannis Tsamardinos, Subramani Mani, and Xenofon D Koutsoukos. Local causal and Markov blanket induction for causal discovery and feature selection for classification part i: Algorithms and empirical evaluation. *Journal of Machine Learning Research*, 11(Jan): 171–234, 2010. 2, 10, 15, 17

[3] Weiwei Liu, Ivor W Tsang, and Klaus-Robert Müller. An easy-to-hard learning paradigm for multiple classes and multiple labels. *The Journal of Machine Learning Research*, 18(1):3300–3337, 2017. 2

[4] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988. ISBN 0-934613-73-7. 7, 9

[5] Thomas Verma and Judea Pearl. Equivalence and synthesis of causal models. In *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, UAI '90, pages 255–270, New York, NY, USA, 1991. Elsevier Science Inc. ISBN 0-444-89264-8. 9

[6] Timo JT Koski and John Noble. A review of Bayesian networks and structure learning. *Mathematica Applicanda*, 40(1):51–103, 2012. 9, 18, 25, 51

[7] David Heckerman. Bayesian networks for data mining. *Data mining and knowledge discovery*, 1(1):79–119, 1997. 9

[8] David Maxwell Chickering, David Heckerman, and Christopher Meek. Large-sample learning of Bayesian networks is NP-hard. *Journal of Machine Learning Research*, 5(Oct):1287–1330, 2004. 9

[9] Tomi Silander and Petri Myllymäki. A simple approach for finding the globally optimal Bayesian network structure. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, pages 445–452. AUAI Press, 2006. 10, 20

[10] James Cussens. Bayesian network learning with cutting planes. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI 2011)*, pages 153–160. AUAI Press, 2011. 10, 60

[11] Mark Barlett and James Cussens. Advances in Bayesian network learning using integer programming. In *Uncertainty in Artificial Intelligence*, page 182, 2013. 10

[12] C Chow and Cong Liu. Approximating discrete probability distributions with dependence trees. *IEEE transactions on Information Theory*, 14(3):462–467, 1968. 10

[13] M Scutari. Learning Bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35(3):1–22, 2010. 10, 21, 30, 31, 49, 58

[14] Peter Spirtes, Clark N Glymour, and Richard Scheines. *Causality from probability*, volume 112. Carnegie-Mellon University, Laboratory for Computational Linguistics, 1989. 10, 17

[15] Peter Spirtes, Clark N Glymour, and Richard Scheines. *Causation, prediction, and search*, volume 81. MIT press, 2000. 10, 17, 62, 67

[16] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, 2016. 10

[17] Gideon Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978. 11, 31

[18] Wray Buntine. Theory refinement on Bayesian networks. In *Proceedings of the Seventh conference on Uncertainty in Artificial Intelligence*, pages 52–60. Morgan Kaufmann Publishers Inc., 1991. 11, 31

[19] David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554, 2002. 11, 31

[20] Ioannis Tsamardinos, Laura E Brown, and Constantin F Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78, 2006. 11, 15, 31, 51, 59

[21] Hoifung Poon and Pedro M. Domingos. Sum-Product Networks: A new deep architecture. In *UAI 2011, Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence, Barcelona, Spain, July 14-17, 2011*, pages 337–346, 2011. 11, 12, 13, 65, 68, 75

[22] Antonio Vergari, Nicola Di Mauro, and Floriana Esposito. Simplifying, regularizing and strengthening Sum-Product Network structure learning. In *Machine Learning and Knowledge Discovery in Databases*, pages 343–358. Springer, 2015. 13, 70

[23] James Martens and Venkatesh Medabalimi. On the expressive efficiency of Sum Product Networks. *CoRR*, abs/1411.7717, 2014. 13

[24] Olivier Delalleau and Yoshua Bengio. Shallow vs. deep Sum-Product Networks. In *Advances in Neural Information Processing Systems 24*, pages 666–674. Curran Associates, Inc., 2011. 13

[25] Dimitris Margaritis. Learning Bayesian network model structure from data. Technical report, Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science, 2003. 15, 17

[26] Ioannis Tsamardinos, Constantin F Aliferis, Alexander R Statnikov, and Er Statnikov. Algorithms for large scale Markov blanket discovery. In *FLAIRS conference*, volume 2, pages 376–380, 2003. 15, 17, 19

[27] Teppo Niinimäki and Pekka Parviainen. Local structure discovery in Bayesian networks. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, pages 634–643. AUAI Press, 2012. 15, 16, 20, 30

[28] Tian Gao, Kshitij Fadnis, and Murray Campbell. Local-to-global Bayesian network structure learning. In *International Conference on Machine Learning*, pages 1193–1202, 2017. 15, 20, 21, 22, 23

[29] Jianjun Zhao and Shen-Shyang Ho. Improving Bayesian network local structure learning via data-driven symmetry correction methods. *International Journal of Approximate Reasoning*, 107:101–121, 2019. 16

[30] Marco Scutari et al. Bayesian network constraint-based structure learning algorithms: Parallel and optimized implementations in the bnlearn R package. *Journal of Statistical Software*, 77(i02), 2017. 17, 19, 28, 31

[31] David Maxwell Chickering. A transformational characterization of equivalent Bayesian network structures. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 87–98. Morgan Kaufmann Publishers Inc., 1995. 19

[32] Christopher Meek. Causal inference and causal explanation with background knowledge. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 403–410. Morgan Kaufmann Publishers Inc., 1995. 19

[33] José A Gámez, Juan L Mateo, and José M Puerta. Learning Bayesian networks by hill climbing: efficient methods based on progressive restriction of the neighborhood. *Data Mining and Knowledge Discovery*, 22(1-2):106–148, 2011. 22

[34] Jacinto Arias, José A Gámez, and José M Puerta. Structural learning of Bayesian networks via constrained hill climbing algorithms: Adjusting trade-off between efficiency and accuracy. *International Journal of Intelligent Systems*, 30(3):292–325, 2015. 23, 31

[35] Luis M de Campos and Javier G Castellano. Bayesian network learning algorithms using structural restrictions. *International Journal of Approximate Reasoning*, 45(2):233–254, 2007. 24

[36] David Heckerman, Dan Geiger, and David M Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20(3):197–243, 1995. 25, 54

[37] Tian Gao and Qiang Ji. Efficient score-based Markov blanket discovery. *International Journal of Approximate Reasoning*, 80:277–293, 2017. 26

[38] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2018. 30

[39] Markus Kalisch, Martin Mächler, Diego Colombo, Marloes H Maathuis, Peter Bühlmann, et al. Causal inference using graphical models with the R package pcalg. *Journal of Statistical Software*, 47(11):1–26, 2012. 31

[40] Joseph Ramsey, Madelyn Glymour, Ruben Sanchez-Romero, and Clark Glymour. A million variables and more: the fast greedy equivalence search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images. *International journal of data science and analytics*, 3(2):121–129, 2017. 31

[41] Bart Selman, Henry A Kautz, and Bram Cohen. Local search strategies for satisfiability testing. *Cliques, Coloring, and Satisfiability: 2nd DIMACS Implementation Challenge*, pages 521–531, 1993. 48

[42] James Cussens. Bayesian network learning by compiling to weighted MAX-SAT. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, pages 105–112. AUAI Press, 2008. 48, 59

[43] Andrew S Fast. *Learning the structure of Bayesian networks with constraint satisfaction*. PhD thesis, University of Massachusetts Amherst, 2010. 50, 87

[44] Tom Claassen and Tom Heskes. A Bayesian approach to constraint based causal inference. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, pages 207–216. AUAI Press, 2012. 51

[45] Kari Markus Juhani Rantanen, Antti Juhani Hyttinen, Matti Juhani Järvisalo, et al. Learning optimal causal graphs with exact search. In *Proceedings of the Ninth International Conference on Probabilistic Graphical Models*. Journal of Machine Learning Research, 2018. 51

[46] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012. 53

[47] Robert E Kass and Adrian E Raftery. Bayes factors. *Journal of the american statistical association*, 90(430):773–795, 1995. 53

[48] Kazuki Natori, Masaki Uto, Yu Nishiyama, Shuichi Kawano, and Maomi Ueno. Constraint-based learning Bayesian networks using Bayes factor. In *Workshop on Advanced Methodologies for Bayesian Networks*, pages 15–31. Springer, 2015. 53, 54

[49] Raanan Yehezkel and Boaz Lerner. Bayesian network structure learning by recursive autonomy identification. *Journal of Machine Learning Research*, 10 (Jul):1527–1570, 2009. 53

[50] Kazuki Natori, Masaki Uto, and Maomi Ueno. Consistent learning Bayesian networks with thousands of variables. In *Advanced Methodologies for Bayesian Networks*, pages 57–68, 2017. 53, 54

[51] Gregory F Cooper and Edward Herskovits. A Bayesian method for constructing Bayesian belief networks from databases. In *Uncertainty Proceedings 1991*, pages 86–94. Elsevier, 1991. 54

[52] Joe Suzuki. A theoretical analysis of the BDeu scores in Bayesian network structure learning. *Behaviormetrika*, 44(1):97–116, 2017. 54

[53] Ruben Martins, Vasco Manquinho, and Inês Lynce. Open-WBO: A modular MaxSAT solver. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 438–445. Springer, 2014. 59

[54] Eric Perrier, Seiya Imoto, and Satoru Miyano. Finding optimal Bayesian network given a super-structure. *Journal of Machine Learning Research*, 9 (Oct):2251–2286, 2008. 62

[55] Sérgio Rodrigues De Morais and Alex Aussem. An efficient and scalable algorithm for local Bayesian network structure discovery. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 164–179. Springer, 2010. 62

[56] Maxime Gasse, Alex Aussem, and Haytham Elghazel. An experimental comparison of hybrid algorithms for Bayesian network structure learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 58–73. Springer, 2012. 63

[57] Mauro Scanagatta, Cassio P de Campos, Giorgio Corani, and Marco Zaffalon. Learning Bayesian networks with thousands of variables. In *Advances in neural information processing systems*, pages 1864–1872, 2015. 63

[58] Alexandru Niculescu-Mizil and Rich Caruana. Inductive transfer for Bayesian network structure learning. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics, AISTATS 2007, San Juan, Puerto Rico, March 21-24, 2007*, pages 339–346, 2007. 65, 67

[59] Diane Oyen and Terran Lane. Transfer learning for Bayesian discovery of multiple Bayesian networks. *Knowl. Inf. Syst.*, 43(1):1–28, April 2015. ISSN 0219-1377. doi: 10.1007/s10115-014-0775-6. 67

[60] Jie Lu, Vahid Behbood, Peng Hao, Hua Zuo, Shan Xue, and Guangquan Zhang. Transfer learning using computational intelligence. *Know.-Based Syst.*, 80(C):14–23, May 2015. ISSN 0950-7051. doi: 10.1016/j.knosys.2015.01.010. 65, 66

[61] Han Zhao, Mazen Melibari, and Pascal Poupart. On the relationship between Sum-Product Networks and Bayesian networks. In *International Conference on Machine Learning*, pages 116–124, 2015. 65

[62] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Trans. on Knowl. and Data Eng.*, 22(10):1345–1359, October 2010. ISSN 1041-4347. doi: 10.1109/TKDE.2009.191. 65, 67, 78

[63] Jianjun Zhao and Shen-Shyang Ho. Structural knowledge transfer for learning Sum-Product Networks. *Knowledge-Based Systems*, 122:159–166, 2017. 66

[64] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 193–200, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-793-3. doi: 10.1145/1273496.1273521. 66

[65] Daniel G. Shapiro, Hector Muñoz-Avila, and David J. Stracuzzi. The special issue of AI magazine on structured knowledge transfer. *AI Magazine*, 32(1): 12–14, 2011. 67

[66] Lilyana Mihalkova and Raymond Mooney. Transfer learning with Markov logic networks. In *Proceedings of the ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning*, Pittsburgh, PA, June 2006. 67

[67] Roger Luis, Luis Enrique Sucar, and Eduardo F. Morales. Inductive transfer for learning Bayesian networks. *Machine Learning*, 79(1-2):227–255, 2010. 67

[68] Robert Gens and Pedro Domingos. Discriminative learning of Sum-Product Networks. In *Advances in Neural Information Processing Systems*, pages 3248–3256, 2012. 67

[69] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems 27*, pages 3320–3328. Curran Associates, Inc., 2014. 68, 83

[70] Amirmohammad Rooshenas and Daniel Lowd. Learning Sum-Product Networks with direct and indirect variable interactions. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 710–718, 2014. 70, 76

[71] Robert Gens and Pedro M. Domingos. Learning the structure of Sum-Product Networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 873–880, 2013. 70, 71, 75, 76, 77, 78

[72] B Woolf. The log likelihood ratio test (the g-test); methods and tables for tests of heterogeneity in contingency tables. *Annals of human genetics*, 21(4):397–409, June 1957. ISSN 0003-4800. doi: 10.1111/j.1469-1809.1972.tb00293.x. 71

[73] B. Iglewicz and D.C. Hoaglin. *How to Detect and Handle Outliers*. ASQC basic references in quality control. ASQC Quality Press, 1993. ISBN 9780873892476. 73

[74] Daniel Lowd and Jesse Davis. Learning Markov network structure with decision trees. In *ICDM*, pages 334–343. IEEE Computer Society, 2010. ISBN 978-0-7695-4256-0. 75

[75] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009. 88

[76] Pradeep Ravikumar, Martin J Wainwright, John D Lafferty, et al. High-dimensional ising model selection using $l1$-regularized logistic regression. *The Annals of Statistics*, 38(3):1287–1319, 2010. 88

[77] Lilyana Mihalkova, Tuyen Huynh, and Raymond J Mooney. Mapping and revising Markov logic networks for transfer learning. In *Aaai*, volume 7, pages 608–614, 2007. 88

[78] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62(1-2):107–136, 2006. 88

[79] Jesse Davis and Pedro Domingos. Deep transfer via second-order Markov logic. In *Proceedings of the 26th annual international conference on machine learning*, pages 217–224. ACM, 2009. 88