MONASH University

Computing Divergences between High Dimensional Graphical Models

Loong Kuan Lee

Doctor of Philosophy

A Thesis Submitted for the Doctor of Philosophy Degree at
Monash University in 2023
Faculty of Information Technology

Abstract

When data are sampled from a probability distribution, and this distribution changes, models learned from the original distribution may degrade in performance when applied to the distribution after the change. This phenomenon is known as concept drift. The development of machine learning methods to adapt to these changes is an active field of research. Consequently, there has also been work done in developing experimental frameworks, namely recovery analysis, to assess the ability of adaptation methods to maintain the quality and generalisation performance of the model in the face of concept drift.

However, the existing works in adaptation assessment do not control for the magnitude of distributional change, or in other words, the divergence between the distribution before and after concept drift. Failure to account for the magnitude of drift in different experiments may be a serious confounding factor that undermines the utility of conclusions that are drawn. To tackle this issue, we will propose methods to compute the divergence between discrete chordal Markov random fields, i.e. decomposable models. These can then be used to create, potentially high dimensional, diescrete distributions that are of a specified amount of divergence away from each other. These differing discrete distributions with divergences of known magnitude can then be used in experimental analyses to understand how alternative methods are affected by drift magnitude.

Specifically, we first show that computing any $\alpha\beta$-divergence between the joint distribution of 2 decomposable models can be reduced to a problem of variable elimination over factors obtained from the divergence we wish to compute. Therefore, the complexity of this computation will be exponential with respect to the size of the largest maximal clique in the decomposable models. We then extend this work to computing the divergence between marginal distributions and conditional distributions obtained from 2 decomposable models. The resulting complexity of these extensions are exponential with respect to the size of the largest Markov blanket over the variables not in the marginal distribution, and the "target" variables in the conditional distribution respectively.

Finally, we propose a method to take an existing decomposable model and modify its parameters to create a new model of a certain "distance" away from the original model. This initial decomposable model can either be learnt from some dataset, or be synthetically generated. Therefore, we will also briefly discuss how to randomly generate synthetic decomposable models with a limit on the model's treewidth.

Declaration of Authorship

This thesis is an original work of my research and contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Signature: _____

Print Name: _____

Date: _____

Acknowledgements

I would like to thank my superiors Geoff Webb, Daniel Schmidt, and Nico Piatkowski for their guidance and advice throughout my PhD, as well as François Petitjean for being my co-supervisor during the start of my PhD.

This research was supported by an Australian Government Research Training Program (RTP) Scholarship.

Contents

List of Figures

List of Tables

List of Definitions

List of Theorems

Chapter 1

Introduction

Computing the divergence between two probability distributions is an operation that has many applications in the fields of machine learning and statistics. For instance, it can be used to estimate the divergence between the underlying distributions of two data samples which is useful in the detection of anomalous regions in spatio-temporal data (Barz et al. 2019) and in various tasks related to the retrieval, classification, and visualisation of time series data (Chen et al. 2020). More importantly, the ability to compute the divergence between high-dimensional distributions is useful in the analysis of and adaptation to concept drift (Schlimmer & Granger, 1986; Sebastião & Gama, 2007; Sebastião et al. 2010; Ditzler & Robi Polikar, 2011; Balzanella et al. 2013; Webb et al. 2018; Goldenberg & Webb, 2019; Baidari & Honnikoll, 2021).

Throughout this thesis, we focus on discrete high-dimensional distributions. These distributions are high-dimensional in that their direct representations, i.e. as a probability vector over all its possible outcomes, are too large to be stored in memory. Furthermore, any computation on these distributions that has a complexity linear to the length of the probability vector, such as the divergence between two distributions, would be intractable as well.

Therefore, this thesis seeks to develop methods for analysing concept drift in high dimensions and for producing synthetic data useful in the assessment of methods that can adapt to concept drift. This work will endeavor to show how the development of these methods can be achieved with the ability to compute divergences between high-dimensional distributions. Therefore, this thesis will also seek to develop tractable methods to compute the divergence between two high-dimensional distributions.

1.1 Motivation: Concept Drift

Most machine learning systems assume that the distribution of the training data is representative of the distribution of future data on which we might want to

carry out some inference task. However, this assumption is not always true. In the real world, the distribution we wish to model might change with respect to some attribute, such as geographic locations, differing populations, and, more commonly, time. These changes can result in the degradation of the performance of a machine learning system. In general, we call distributional changes over some general factor *concept shift* (Webb et al. 2018). On the other hand, due to how widespread of a problem distributional changes over time are, this type distributional change goes by different names in different fields, such as *change points* (Carrera & Boracchi, 2018) and *concept drift* (Schlimmer & Granger, 1986). For ease of exposition, in this thesis we will use *concept drift* to refer to any type of distribution change, whether it be over time or over some other factor. Furthermore, we only focus on discrete distributions throughout this thesis.

Concept drift can be characterised using various qualitative attributes and quantitative measures (Webb et al. 2016). One quantitative measure is the magnitude of change in the distribution due to the occurrence of concept drift. This quantitative measure is also known as the magnitude of concept drift, or drift magnitude (Webb et al. 2016). Specifically, we can use the divergence between the distribution before and after drift as the measure of the magnitude of change in the distribution.

Furthermore, over the past decade, there has been an increase in the application of machine learning systems on high-dimensional data streams. There has been a significant amount of recent work into detecting and adapting to high-dimensional concept drift (Zimek et al. 2012; David Destephen Lavaire et al. 2015; Alippi et al. 2016; Sethi & Kantardzic, 2017; Boracchi et al. 2018; W. Zhang et al. 2019; Carrera, 2020). In order to test these drift detection and adaptation methods, we require datasets containing concept drift. There are 3 types of datasets commonly used in testing drift detection techniques:

- Real world datasets that are suspected to have concept drift (Harries, 1999; Gonçalves Jr & Barros, 2013). However, one issue is that there are relatively few datasets for which we definitively know that both concept drift has occurred and the characteristics of these drifts.
- Synthetic datasets with concept drift (Widmer & Kubat, 1996; Street & Kim, 2001; Minku et al. 2010). Although we are able to control the occurrences and some characteristics of concept drift in synthetic datasets, it is arguable that these datasets might not have inter-variable relationships that are realistic. Furthermore, while it might be possible to increase or decrease the magnitude of drift between each generated dataset, it is not possible to control by how much the drift magnitude differs between these datasets.
- Modification of real world datasets to inject synthetic concept drift (Bifet et al. 2009a; Bifet et al. 2009b; Klinkenberg, 2001; Scholz & Klinkenberg,

FIGURE 1.1: Spectrum of how "synthetic" or "real" datsets containing concept drift are, and where the method we wish to develop lies in this spectrum.

2005; Robert Polikar et al. 2001; Žliobaite, 2010; Carrera & Boracchi, 2018). This represents a hybrid approach to using real world data and synthetic datasets. This approach has been used before to carry out a study on how well various drift adaptation techniques recover from drift, also known as recovery analysis (Shaker & Hüllermeier, 2015). Modifying datasets to achieve a specified magnitude of concept drift requires a method to compute the divergence between the underlying distribution of the data before and after drift. However, since the actual underlying distributions of the dataset being modified are unknown, only an estimation of the divergence between these underlying distributions can be obtained, which is sub-optimal when creating datasets meant for assessing concept drift adaptation techniques.

When carrying out experimental analysis like recovery analysis (Shaker & Hüllermeier, 2015) on different concept drift adaptation techniques, the magnitude of drift is an important confounding factor that can have varying effects depending on the adaptation technique. For instance, a reasonable hypothesis to have is that methods that try to detect concept drift explicitly might adapt quicker to high magnitudes of concept drift compared to methods that adapt to drift passively via some forgetting mechanism. Therefore, it will be insightful to carry out experiments like recovery analysis whilst controlling for the magnitude of drift present in the datasets used for the analysis.

Consequently, the main motivating factor in this thesis is to develop a method that, in exchange for being more on the synthetic side of the spectrum compared to methods that modify the dataset directly, is able to generate high-dimensional datasets that contain controlled magnitudes of concept drift. However, in order to be able to even approach this problem in the first place, we require a method to compute the divergence between high-dimensional discrete distributions.

1.2 Problem: Computing Divergence between High-Dimensional Distributions

The most straightforward approach to computing the divergence between 2 discrete distributions is to make minimal assumptions regarding the structure of and independencies within these distributions. Using this approach, the distributions are represented as probability vectors over the support of these distributions and the divergence between these probability vectors is computed directly (Sebastião & Gama, 2007; Sebastião et al. 2010; Webb et al. 2018). The problem with this approach is that, for a distribution with n variables, each taking k values, the support size of this distribution in the worst case, when no varlable-value combination has a probability of 0, is k^n, i.e. exponential with respect to the number of variables n. Therefore, both storing, and computing the divergence between, these joint distributions are intractable as n increases. Of course, when these distributions are estimated using the empirical distribution from a finite set of samples, the support size of these estimated distributions will only grow linearly with respect to the size of the sample set (Webb et al. 2018). However, relying on this to reduce the complexity potentially prevents the use of prior distributions or smoothing techniques when estimating the distributions, which can be useful when we have a limited amount of samples. This is because using a prior distribution or some smoothing technique can cause the support of the estimation distributions to scale exponentially with respect to the number of variables, $\mathcal{O}(2^n)$, instead of just linearly with respect to just sample size.

It is possible to approximate, instead of computing exactly, the divergence between two high-dimensional distributions, whose support size is exponential with respect to the number of variables. For instance, we can find an embedding of these high-dimensional distributions into the simplex of a lower dimension that preserves the structure between these distributions given some divergence (Bhattacharya et al. 2009; Abdullah et al. 2016; Goldenberg & Webb, 2020). However, the problem then shifts towards obtaining and storing these high-dimensional distributions while avoiding storage complexity exponential with respect to the number of variables.

Therefore, we will approach this problem by assuming that the structure and independencies in these distributions make them amendable for more compact representations. As we will discuss in Section 2.2, there is a rich body of literature in the field of graphical models for graphically representing independence assumptions and using these independencies to obtain a compact representation of a distribution with these independencies. In fact, work exists on efficiently computing the Kullback-Leibler divergence between discrete distributions represented by Bayesian networks (Moral et al. 2021). It is also possible to tractably compute the Kullback-Leibler divergence between a general Markov network and a Markov network where inference tasks are tractable (Koller & Friedman, 2009).

However, for the purposes of computing the magnitude of concept drift, we might want to use divergences other than the Kullback-Leibler divergence. One reason is that the Kullback-Leibler divergence is not bounded and therefore it can be difficult to compare the Kullback-Leibler divergence across contexts. Furthermore, due to the KL divergence being undefined when $\mathbb{P}(\boldsymbol{x}) > 0$ and $\mathbb{Q}(\boldsymbol{x}) = 0$, there's a possibility of it being undefined in situations where any attribute-value combination, $\boldsymbol{x}$, has zero probability. This situation is likely to occur in sparse data and also when some attribute-value combinations truly do have zero probability. Furthermore, the KL divergence is almost surely to be undefined when the probability of some $\boldsymbol{x}$ goes from having true zero probability, to having a non-zero probability, and vice versa. For instance, the probability of a man being diagnosed as pregnant might be truly zero in the past, but with transgender acceptance and social transitioning being more widespread, the probability of a man being diagnosed as pregnant is currently non-zero. For an example of the probability of some $\boldsymbol{x}$ going from non-zero to true zero, take the probability of finding a certain species of animal in some habitat before and after this species goes extinct.

1.3 Contributions

Motivated by the problems above, the main contributions of this thesis are as follows:

1. Developing a method to compute the $\alpha\beta$-divergence between the joint distributions modelled by 2 high-dimensional probabilistic graphical models.

   Specifically, we develop a general method to compute functionals between the joint distributions of 2 decomposable models which we then apply to the problem of computing the $\alpha\beta$-divergence between these models.

2. Extending the work in computing functionals between joint distributions of decomposable models in order to compute

   2.1. functionals between marginal distributions over some subset of the variables in the decomposable models, and

   2.2. functionals between conditional distributions, where some subset of the variables in the decomposable model are the "target" variables, with the other variables being the condition.

3. Developing a method to generate high-dimensional data containing known magnitudes of concept drift by modifying a given decomposable model.

   In the process we will develop methods to:

3.1. modify the parameters of a decomposable model such that the resulting model is a specific magnitude of $\alpha\beta$-divergence away from the original model, and to

3.2. generate random decomposable models with a given treewidth

1.4 Thesis Structure & Organisation

We will begin this thesis in Chapter 2 with an overview of the background material and notation used throughout this thesis. Past Chapter 2, all material will be new unless stated otherwise. We begin discussing the contributions of this thesis with the core contribution of computing the joint divergence between high-dimensional decomposable models in Chapter 3. We then extend this contribution in Chapters 4 and 5 in order to compute the divergence between marignal and conditional distributions encoded in the decomposable models. Once the contributions related to divergence computation are detailed, we then apply these divergence computation techniques on the problem of modifying the parameters of a decomposable model in order to achieve a certain amount of conditional or joint divergence away from the original model in Chapter 6. We then end the contribution portion of the thesis with Chapter 7, a chapter on using the methods we have developed throughout this thesis to both create and test a method to estimate the divergence between the underlying distribution of two samples. Lastly, we conclude the thesis with Chapter 8, containing a summary of the entire thesis, some closing thoughts, conclusions, and future work that can be done, either thanks to the methods developed in this thesis, or to extend on the work presented in the thesis.

Chapter 2

Background

In this chapter, we will introduce key definitions and background work, upon which, this thesis builds. But first of all, let us define some basic notation and conventions that we will use throughout this thesis. In this thesis we will only consider discrete random variables (rvs) X_v that take values from the state space $\mathcal{X}_v$, i.e. $\text{Dom}(X_v) = \mathcal{X}_v$. Here v represents the label/identifier given to the rv X_v.

Now assume we have a set of labels A with each label identifying a unique rv $\boldsymbol{X} = (X_v : v \in A)$. For a subset of A, $B \subseteq A$, we let $\mathcal{X}_B = \bigotimes_{v \in B} \mathcal{X}_v$ be the Cartesian product of the state spaces of each rv in $\boldsymbol{X}_B$. Then $\boldsymbol{x}_B = (x_v : v \in B)$ are the elements of $\mathcal{X}_B$ with $\boldsymbol{X}_B = (\boldsymbol{X}_v : v \in B)$ being the rvs associated with the labels in B. Furthermore, we allow access to the labels of the rvs using the notation $A_{\boldsymbol{X}_B} = B$.

Additionally, through a slight abuse of notation, we also allow set operations between subsets of the rvs $\boldsymbol{X}$ and subsets of the rv labels A. These set operations, such as $\boldsymbol{X} \setminus B$ and $\boldsymbol{X} \cap B$, are equivalent to the set operations between the labels associated with the subset of $\boldsymbol{X}$ and the subset of the labels. For example $\boldsymbol{X} \setminus B = A \setminus B$ and so on. Furthermore, we shall also the define the general shorthand, where for some set C and D, $C_D = D_C = C \cap D$. This implies that for some $\boldsymbol{Y} \subset \boldsymbol{X}$, $\boldsymbol{X}_{\boldsymbol{Y}} = \boldsymbol{Y}_{\boldsymbol{X}} = \boldsymbol{X} \cap \boldsymbol{Y} = \boldsymbol{Y}$.

Let P be a discrete probability distribution over the rvs $\boldsymbol{X}$. Then for any subset of $\boldsymbol{X}$, $\boldsymbol{Y} \subseteq \boldsymbol{X}$, $P_{\boldsymbol{Y}}$ denotes the marginal distribution of P over $\boldsymbol{Y}$. Furthermore, for some subset of the rv labels A, $B \subseteq A$, P_B denotes the marginal distributon over the rvs associated with the labels in B, i.e. $P_B = P_{\boldsymbol{X}_B}$. When the probability over many elements in $\mathcal{X}$ is zero, it is often convenient to think about the distribution in terms of just $\boldsymbol{x} \in \mathcal{X}$ with a non-zero probability, $P(\boldsymbol{x}) > 0$. We call this set of values from $\mathcal{X}$, $S \subseteq \mathcal{X}$, the support of P.

When dealing with conditional distributions of $\boldsymbol{Y}$ given $\boldsymbol{Z}$, $P_{\boldsymbol{Y}|\boldsymbol{Z}}$, it is sometimes more convenient to use a representation of the conditional distribution without the need of subscripts, $P_{\boldsymbol{Y}|\boldsymbol{Z}} = P(\boldsymbol{Y} \mid \boldsymbol{Z})$. Here the use of rvs $\boldsymbol{Y}$ and $\boldsymbol{Z}$ as "arguments" to P implicitly asserts that P is a functional that takes values from the domain of $\boldsymbol{Y}$ and $\boldsymbol{Z}$, $\mathcal{Y}$ and $\mathcal{Z}$ respectively, which are the same inputs $P_{\boldsymbol{Y}|\boldsymbol{Z}}$ takes. Similarly, we

will sometimes express the conditional distribution, when conditioned on a specific value $z \in \mathcal{Z}$, as $P(\boldsymbol{Y} \mid z)$, instead of the more formally correct expression of $P_{Y|z}$. Finally, when passing specific values $y \in \mathcal{Y}$ and $z \in \mathcal{Z}$ into P, it is implicit that $P(y \mid z) = P_{Y|Z}(y \mid z)$.

2.1 Graph Theoric Background and Notation

A graph, $\mathcal{G} = (V, E)$[1], consists of $n = |V|$ vertices and edges $(u, v) \in E$, where $u \neq v$, i.e. we forbid self-edges in this thesis. The edges in E can either be directed, where $(u, v) \neq (v, u)$, forming a directed graph (DG), or undirected, where $(u, v) = (v, u)$, forming an undirected graph (UG). We will further expand on the particular notaion and characteristics of directed and undirected graphs in Sections 2.1.1 and 2.1.2 respectively. But for now, we will discuss the notation and conventions relating to any graph, directed or undirected, in general.

[1] When dealing with multiple graphs simultaneously, we will use the notation $\mathcal{G}_i = (V(\mathcal{G}_i), E(\mathcal{G}_i)) = (V_i, E_i)$ in order to differentiate the vertex and edge sets of different graphs.

For any subset of the vertices in $\mathcal{G}$, $A \subseteq V$, $\mathcal{G}(A)$ denotes the *induced subgraph* of $\mathcal{G}$ containing only the vertices in A:

$$\mathcal{G}(A) = (A, E(\mathcal{G}(A)))$$
$$E(\mathcal{G}(A)) = \left\{ (u, v) \,\middle|\, (u, v) \in E(\mathcal{G}) \wedge u, v \in A \right\}$$

Furthermore, any vertex v is a neighbour of u if there exists an edge $e \in E$, such that both vertices are in the edge e, i.e. $e = (v, u) \in E$. A path in $\mathcal{G}$ is a sequence of edges in E that joins a sequence of distinct vertices together, forming a chain of vertices. When a sequence of edges joins a sequence of distinct vertices such that it starts and ends at the same vertex, it forms a cycle.

A graph is connected if there exists a path from any vertex to any other vertex in the graph. The opposite of a connected graph is a disconnected graph, which can be viewed as comprising of multiple connected components or subgraphs.

A connected component of a graph $\mathcal{G}$ is a maximal set of vertices A such that the induced subgraph $\mathcal{G}(A)$ contains a path from every vertex in A to every other vertex in A. A being a maximal set means that no other vertices in $\mathcal{G}$ can be added to A while preserving this connectivity property. Therefore, the vertices of any graph can be partitioned into set(s) of connected components, with a connected graph only containing a single connected component.

2.1.1 Directed Graphs (DGs)

A directed graph (DG), $\vec{\mathcal{G}}$, is a graph where its vertices are connected by directed edges, i.e. $\forall (u,v) \in E : (u,v) \neq (v,u)$. For any directed edge $e = (u,v)$, we will say e is directed from u to v, i.e. $u \to v$, and that u is the head of edge e while v is the tail.

As an extension to the undirected definition of paths that do not consider directionality, a *directed* path is a sequence of directed edges, $(e_1, e_2, ...)$, such that the tail of each edge e_i is the head for the next edge e_{i+1}. Similarly, a *directed* cycle is a directed path that starts and ends at the same vertex.

2.1.1.1 Directed Acyclic Graphs (DAGs)

When a DG does not have any directed cycles, it is also known as a directed acyclic graph (DAG). Due to the lack of directed cycles in a DAG, for each vertex $v \in V$, we can categorise its neighbours as either parents or children of the vertex depending on whether the edge is directed from the neighbour to v or from v to the neighbour respectively.

$$\mathrm{pa}\left(v, \vec{\mathcal{G}}\right) := \left\{ u \;\middle|\; u \in V\left(\vec{\mathcal{G}}\right) \setminus \{v\} \wedge (u,v) \in E\left(\vec{\mathcal{G}}\right) \right\}$$
$$\mathrm{ch}\left(v, \vec{\mathcal{G}}\right) := \left\{ u \;\middle|\; u \in V\left(\vec{\mathcal{G}}\right) \setminus \{v\} \wedge (v,u) \in E\left(\vec{\mathcal{G}}\right) \right\}$$

or in other words, $\mathrm{pa}\left(v, \vec{\mathcal{G}}\right)$ and $\mathrm{ch}\left(v, \vec{\mathcal{G}}\right)$ are the sets of parent and child vertices of v in $\vec{\mathcal{G}}$ respectively.

2.1.2 Undirected Graphs (UGs)

As opposed to its directed counterpart, an undirected graph (UG), $\mathcal{G}$, is a graph with undirected edges, i.e. $(u,v) = (v,u)$. When a subset of vertices, $C \subseteq V(\mathcal{G})$, is complete in $\mathcal{G}$, i.e. the vertices in C are connected to each other in $\mathcal{G}$, or more formally when

$$\forall u, v \in C : u \neq v \Rightarrow (u,v) \in E(\mathcal{G})$$

the vertices in C form a clique. When there are no other vertices in V, $v \in V \setminus C$, that can be added to the set of vertices C to form a clique, we call C a *maximal* clique. Furthermore, let $\boldsymbol{C}(\mathcal{G})$ denote the set of maximal cliques in $\mathcal{G}$.

2.1.2.1 Trees

An UG, $\mathcal{T} = (V(\mathcal{T}), E(\mathcal{T}))$, that contains no cycles is also known as a tree. For any subset of vertices $A \subseteq V(\mathcal{T})$, if the induced subgraph of $\mathcal{T}$ on A, $\mathcal{T}(A)$, is connected,

it is also called a subtree of $\mathcal{T}$. A tree $\mathcal{T}$ can be "rooted" by choosing an arbitary vertex in $V(\mathcal{T})$ to be the "root vertex". When a tree is rooted, each edge in $E(\mathcal{T})$ will have an implicit direction from the vertex closest to the root to the vertex further from the root. As such, the neighbours of vertices in a rooted tree can also be categorised into "parent" and "child" vertices, similar to DAGs.

2.1.3 Chordal Graphs

[2] Chordal graphs are sometimes also referred to as triangulated graphs.

An UG $\mathcal{G} = (V, E)$ is chordal[2] if every cycle of length greater than three has a chord, i.e. an edge that is not part of the cycle but connects two vertices of the cycle. Furthermore, any induced subgraph of a chordal graph is also chordal (Blair & Peyton, 1993), which implies that the removal of any vertex from a chordal graph results in yet another chordal graph.

Corollary 1 (Vertex deletion from chordal graph) Let $\mathcal{G}$ be a chordal graph and $v \in V(\mathcal{G})$ be a vertex in $\mathcal{G}$. Then removing vertex v from $\mathcal{G}$ results in a new chordal graph $\mathcal{G}'$.

Proof Assume we want to remove vertex $v \in V(\mathcal{G})$ from chordal graph $\mathcal{G}$, resulting in a new graph $\mathcal{G}'$. Then the set of vertices in $\mathcal{G}'$ will be the set $A = V(\mathcal{G}) \setminus \{v\}$. Since removing vertex v involves removing both vertex v and any edges in $E(\mathcal{G})$ that contain v, the removal of v is equivalent to the induced subgraph on the set of vertices A. Since we know that the induced subgraph of a chordal graph is still chordal (Blair & Peyton, 1993), the graph $\mathcal{G}'$ is chordal as well.

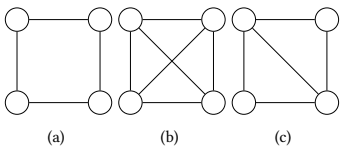


Figure 2.1: (a) a non-chordal graph, (b) a possible triangulation of this graph, (c) a minimal triangulation of this graph.

Any non-chordal graph $\mathcal{G}$ can be converted into a chordal graph by "triangulating" it, i.e. by finding and adding a set of edges to the graph that will make it chordal. Finding the minimal triangulation of a non-chordal graph, i.e. the minimal number of edges to add to a non-chordal graph to make it chordal, is a **NP**-hard problem (Yannakakis, 1981; Heggernes, 2006). Instead, a valid, but not necessarily minimal, triangulation can be found with time polynomial with respect to the number of vertices n (Mezzini & Moscarini, 2010; Berry et al. 2004; Heggernes, 2006). See Figure 2.1 for an example of a non-chordal graph, and both a non-minimal and minimal triangulation of it.

2.1.3.1 Junction Trees/Forest

[3] The clique-intersection property is sometimes also known as the running-intersection property (Koller & Friedman, 2009).

There are ways to characterise chordal graphs other than an UG with no cycles of length greater than three without a chord. One way is as a tree over the maximal cliques $\boldsymbol{C}(\mathcal{G})$, i.e. a clique tree.

Theorem 1 (Junction trees (Blair & Peyton, 1993, Theorem 3.1)) A connected graph $\mathcal{G}$ is chordal if and only if there exists a tree $\mathcal{T} = (\boldsymbol{C}(\mathcal{G}), \boldsymbol{S}(\mathcal{G}))$ for which the clique-intersection property[3] holds. Or in other words, for every pair of distinct maximal cliques $C_1, C_2 \in \boldsymbol{C}(\mathcal{G})$, the vertices in the set $C_1 \cap C_2$ are contained in every maximal clique on the path connecting C_1 and C_2 in $\mathcal{T}$.

Clique trees that satisfy the clique-intersection property are also known as junction trees, with the set of edges $\boldsymbol{S}(\mathcal{G})$ in the junction tree being known as the minimal separators of chordal graph $\mathcal{G}$ (Cowell et al. 1999, sec. 4.3). Of course, it might be possible for $\mathcal{G}$ to not be a connected graph, but a disconnected one instead. In such cases, $\mathcal{G}$ won't have a single junction tree, but multiple cliques trees for each connected component, i.e. a junction forest.

Definition 1 (Junction forest) Let $\mathcal{G}$ be a disconnected chordal graph where each connected component/subgraph is a chordal graph itself. Then the junction forest of $\mathcal{G}$ is the collection of junction trees $\boldsymbol{\mathcal{T}} = \{\mathcal{T}_1, \mathcal{T}_2, ...\}$, one for each connected chordal subgraph in $\mathcal{G}$.

2.1.3.2 Chordal Graphs as Intersections of Subtrees

Another way to characterise chordal graphs is with *subtree graphs*.

Definition 2 (Subtree graph) Let T be a tree with subtrees $F = \{v_1, \dots, v_n\}$, i.e. n connected subgraphs of T. Then the subtree graph of the family of subtrees F is the intersection graph, $\mathcal{G}$, of F, where each subtree in F is a vertex in $\mathcal{G}$ and an edge between vertex associated with subtrees v_i and v_j exists if v_i and v_j intersect in T, $v_i \cap V_j \neq \emptyset$.

For example, in Figure 2.2a we have a tree T_a with 6 vertices and 5 subtrees. The following is a list of how the different subtrees over T_a intersect with each other:

1. v_1 intersects with v_2
2. v_1 intersects with v_3
3. v_2 intersects with v_3
4. v_3 intersects with v_4
5. v_3 intersects with v_5
6. v_4 intersects with v_5

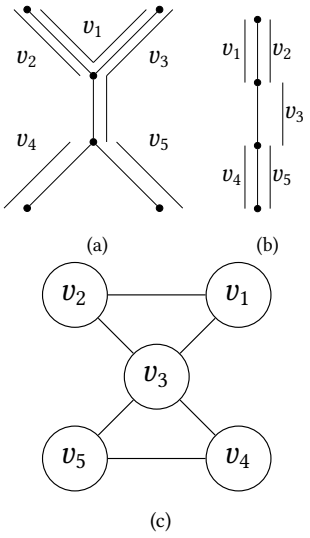


Figure 2.2: (a,b) 5 subtrees over a single tree, and (b) the intersection graph over the subtrees.

These intersections result in the subtree graph, or intersection graph, in Figure 2.2c. In fact, different subtree families can result in the same subtree/chordal graph. For example, both subtree families in Figures 2.2a and 2.2b results in the subtree graph in Figure 2.2c.

More importantly, we can observe that the subtree graph in Figure 2.2c is also a chordal graph. This is not a coincidence as this is true for any subtree graph.

Theorem 2 (Subtree and chordal graphs, Gavril (1974, Theorem 3)) A graph $\mathcal{G}$ is a subtree graph if and only if it is a chordal graph.

2.2 Graphical Models

Consider the problem of representing the joint distribution, $\mathbb{P}_X$, over the discrete random variables $\boldsymbol{X} = (X_1, \dots, X_n)$. Even assuming binary variables, naively representing the probability of each possible outcome over n binary variables will require the use of $2^n - 1$ parameters. More generally, to naively represent the joint distribution over n variables in this manner, we require an exponential number of parameters with respect to n, which for large values of n, is impractical (Pearl, 1988, p. 78).

This naive method of representing the joint distribution makes no use of any additional knowledge we may have of the relationship between the random variables in $\boldsymbol{X}$. However, consider the case where we do know that the variables in $\boldsymbol{X}$ are mutually independent.

Definition 3 (Mutual Independence) Let $\boldsymbol{X} = (X_1, \dots, X_n)$ be random variables with joint probability mass function (pmf) $\mathbb{P}_X(x_1, \dots, x_n)$ and marginal pmfs $\mathbb{P}_{X_i}(x_i)$. Then, the variables $\boldsymbol{X}$ are called mutually independent random variables if, for every $(x_1, \dots, x_n) \in \text{Dom}(X_1) \times \dots \times \text{Dom}(X_n)$:

$$\mathbb{P}(x_1, \dots, x_n) = \prod_{i=1}^{n} \mathbb{P}_{X_i}(x_i) \tag{2.1}$$

From Definition 3, we know that in order to represent the joint pmf of n mutually independent random variables, we only need to represent the marginal distribution over each individual random variable. Let k be the maximum cardinality of any random variable in $\boldsymbol{X}$, i.e. $k = \max(|\text{Dom}(X_1)|, \dots, |\text{Dom}(X_n)|)$. Then we will only need at most k parameters to represent each of the n marginal pmfs and therefore $n \cdot k$ parameters to represent the joint pmf over these variables. This is a significant reduction in the number of parameters when compared to the our initial naive example.

Therefore, knowledge regarding the relationships between random variables can be used to produce more compact representations of the joint pmf over these variables. However, as stated by Koller & Friedman (2009):

> While independence is a useful property, it is not often that we encounter two independent events. A more common situation is when two events are independent given additional events.

This more common situation is also known as conditional independence. The notion of conditional independence was systematically studied in Dawid (1979), whose notation we will use, and Dawid (1980) provided a formal treatment to the subject.

Definition 4 (Conditional Independence) Let $X \perp\!\!\!\perp Y \mid Z$ denote that random variables X and Y are conditionally independent given variables Z. Then the following are equivalent for all assignments of $x \in \text{Dom}(X)$, $y \in \text{Dom}(Y)$, and $z \in \text{Dom}(Z)$ when $\mathbb{P}_Z(z) > 0$:

$$\begin{aligned} X \perp\!\!\!\perp Y \mid Z &\Leftrightarrow \mathbb{P}(x, y \mid z) = \mathbb{P}(x \mid z)\mathbb{P}(y \mid z) \\ &\Leftrightarrow \mathbb{P}(x \mid y, z) = \mathbb{P}(x \mid z) \end{aligned}$$

From Definition 4, we can immediately observe that exploiting conditional independence allows us to reduce the complexity of representing conditional distributions, either via factorisation into smaller conditional distributions, or by removing conditional variables that are irrelevant.

It is possible to represent the notion of $X \perp\!\!\!\perp Y \mid \boldsymbol{Z}$ in both UGs and DAGs. Both these graphical representations are the underlying graph structures for Markov networks (MNs) and Bayesian networks (BNs) respectively.

Definition 5 (I-map) Let $\mathcal{K}$ be any graph object associated with a set of independencies $\mathcal{I}(\mathcal{K})$. We say $\mathcal{K}$ is an I-map for a set of independencies $\mathcal{I}$ if $\mathcal{I}(\mathcal{K}) \subseteq \mathcal{I}$ (Koller & Friedman, 2009).

We will provide the relevant details regarding BNs and MNs in Subsections 2.2.2 and 2.2.1 respectively. Then in Subsection 2.2.3, we will provide details on how BNs and MNs are related to one another.

2.2.1 Bayesian Network

The graphical structure of a BN is a directed acyclic graph (DAG) as described in Section 2.1.1.1. Such a DAG encodes conditional independencies by both its own notion of paths[4] and with a criterion called d-separation (Pearl et al. 1989).

[4] recall that we define a path in a directed graph to be a sequence of adjacent edges, ignoring the directions of these edges.

Definition 6 (d-separation (Pearl et al. 1989)) Let $\boldsymbol{v}$, $\boldsymbol{u}$, and $\boldsymbol{w}$ be three disjoint subsets of vertices in a DAG $\vec{\mathcal{G}}$. Then $\boldsymbol{w}$ d-separates $\boldsymbol{v}$ from $\boldsymbol{u}$ if and only if there is no path from a vertex in $\boldsymbol{v}$ to a vertex in $\boldsymbol{u}$, along which the following 2 conditions hold:

1. every vertex with converging arrows either is or has a descendent in $\boldsymbol{w}$, and
2. every other vertex in the path is outside $\boldsymbol{w}$.

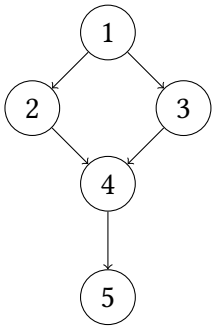


Figure 2.3: Example of a DAG.

Therefore, any DAG will also implicitly provide a list of conditional independencies, $\mathcal{I}(\vec{\mathcal{G}})$, encoded by d-separation of vertices in $\vec{\mathcal{G}}$. For example, in Figure 2.3 the vertices $\{2\}$ and $\{3\}$ are d-separated by $\{1\}$, i.e. $X_2 \perp\!\!\!\perp X_3 \mid X_1$ since:

1. the path $2 \leftarrow 1 \rightarrow 3$ only contains vertices in $\{1\}$
2. the path $2 \rightarrow 4 \leftarrow 3$ only contains vertices with converging arrows but is not in, nor have descendents in, $\{1\}$.

However 2 and 3 are *not* d-separated by $\{1, 5\}$ since:

1. the path $2 \rightarrow 4 \leftarrow 3$ only contain vertices with converging arrows and have descendents that are in $\{1, 5\}$

Consequently, the DAG in Figure 2.3 encodes the conditional independence $X_2 \perp\!\!\!\perp X_3 \mid X_1$, but *not* the conditional independence $X_2 \perp\!\!\!\perp X_3 \mid X_{1,5}$. These conditional independencies in a DAG encoded by d-separation can then be used to factorise a distribution that $\mathcal{I}(\vec{\mathcal{G}})$ is an I-map of.

Theorem 3 (Factorising $\mathbb{P}$ (Koller & Friedman, 2009, Theorem 3.1)) Let $\vec{\mathcal{G}}$ be a DAG over vertices V, where each variable in $\boldsymbol{X}$ is associated with a vertex in v, $\boldsymbol{X} = \{X_v \mid v \in V\}$. Furthermore, let $\mathbb{P}$ be a joint distribution over the variables $\boldsymbol{X}$. Then, if $\mathcal{I}(\vec{\mathcal{G}})$ is an I-map of $\mathbb{P}$, $\mathcal{I}(\vec{\mathcal{G}}) \subseteq \mathcal{I}(\mathbb{P})$, then $\mathbb{P}$ factorises acording to $\vec{\mathcal{G}}$ as such:

$$\mathbb{P}_X(\boldsymbol{x}) = \prod_{i \in V(\vec{\mathcal{G}})} \mathbb{P}(X_i \mid \boldsymbol{X}_{\mathrm{pa}(i,\vec{\mathcal{G}})})$$

where $\mathbb{P}(i, \vec{\mathcal{G}})$ is the set of parent vertices to vertex i in $\vec{\mathcal{G}}$.

Therefore, a BN is a pair $\mathcal{B} = (\vec{\mathcal{G}}, \mathbb{P})$ where $\mathbb{P}$ factorises over the DAG $\vec{\mathcal{G}}$, and where $\mathbb{P}$ is specified as a set of conditional probability tables (CPTs) for each vertex in

$\vec{\mathcal{G}}$ (Koller & Friedman, 2009, Definition 3.5). For example, the DAG in Figure 2.3 allows for the following decomposition of $\mathbb{P}_X$:

$$\mathbb{P}(\boldsymbol{x}) = \mathbb{P}(x_1) \cdot \mathbb{P}(x_2 \mid x_1) \cdot \mathbb{P}(x_3 \mid x_1) \cdot \mathbb{P}(x_4 \mid x_{2,3}) \cdot \mathbb{P}(x_5 \mid x_5)$$

which reduces the complexity for storing the parameters of the distribution $\mathbb{P}$ from being $\mathcal{O}(2^5)$ to $\mathcal{O}(2^3)$.

2.2.2 Markov Network

Recall from Section 2.1.2 that an undirected graph (UG), $\mathcal{G} = (V(\mathcal{G}), E(\mathcal{G}))$[5], consists of $n = |V(\mathcal{G})|$ vertices connected by undirected edges $(u, v) = (v, u) \in E(\mathcal{G})$. When a subset of vertices, $C \subseteq V(\mathcal{G})$, are fully connected, they form a clique, Furthermore, let $\hat{\boldsymbol{C}}(\mathcal{G})$ and $\boldsymbol{C}(\mathcal{G})$ denote the set of all cliques and maximal cliques in $\mathcal{G}$ respectively. Let each vertex $v \in V(\mathcal{G})$ uniquely identify the rv X_v, $\boldsymbol{X} = (X_v : v \in V(\mathcal{G}))$. Then we can define 3 properties over $\mathcal{G}$ that each encode different types of independencies between the variables in $\boldsymbol{X}$:

[5] this somewhat odd notation for the vertex and edge set of a graph will be useful in later chapters when we are dealing with multiple graphs simultaneously.

Definition 7 (Markov properties) Let $\mathcal{G} = (V(\mathcal{G}), E(\mathcal{G}))$ be an UG with associated rvs $(X_v : v \in V(\mathcal{G}))$. Then we can define 3 Markov properties that represent 3 different types of independencies encoded within the structure of $\mathcal{G}$. Therefore, we will define the following Markov properties in terms of their associated set of independencies:

$\mathcal{I}_p$ the pairwise Markov property

$$\mathcal{I}_p(\mathcal{G}) = \{(X_u \perp\!\!\!\perp X_v \mid \boldsymbol{X}_{V(\mathcal{G})\setminus\{u,v\}}) : (u, v) \notin E(\mathcal{G})\}$$

$\mathcal{I}_l$ the local Markov property

$$\mathcal{I}_l(\mathcal{G}) = \{(X_v \perp\!\!\!\perp X_{V(\mathcal{G})\setminus\mathcal{N}_{\mathcal{G}}(v)\cup\{v\}} \mid \boldsymbol{X}_{\mathcal{N}_{\mathcal{G}}(v)}) : v \in V(\mathcal{G})\}$$

where $\mathcal{N}_{\mathcal{G}}(v)$ is the neighbours of vertex v, i.e. the set of vertices that are connected to v, $\mathcal{N}_{\mathcal{G}}(v) = \{u : (v, u) \in E(\mathcal{G})\}$

$\mathcal{I}_g$ the global Markov property

$$\mathcal{I}_g(\mathcal{G}) = \{\boldsymbol{X}_A \perp\!\!\!\perp \boldsymbol{X}_B \mid \boldsymbol{X}_S : (A, B, S \subseteq V(\mathcal{G})) \wedge \text{sep}_{\mathcal{G}}(A; B \mid S)\}$$

where $\text{sep}_{\mathcal{G}}(A; B \mid S)$ denotes that the set of vertices S separates the vertices in A and B in $\mathcal{G}$, i.e. there is no path from the vertices in A to the vertices in B that does not pass through S in $\mathcal{G}$.

A pmf $\mathbb{P}$ on $\mathcal{X}_{V(\mathcal{G})}$ is said to obey a Markov property if the property's respective set of independencies is an I-map of the independencies in $\mathbb{P}$.

The local Markov property enforces that X_v is independent of everything else given the random variables of its neighbours $(X_u : u \in \mathcal{N}_{\mathcal{G}}(v))$. Due to this, $\mathcal{N}_{\mathcal{G}}(v)$ is also known as the Markov blanket of v in graph $\mathcal{G}$. On the other hand, the global Markov property is important because it gives a general criterion for deciding when 2 disjoint sets of random variables, $\boldsymbol{X}_A$ and $\boldsymbol{X}_B$, are conditionally independent given another disjoint set of variables $\boldsymbol{X}_S$ (Lauritzen, 1996).

More importantly, if we assume a pmf $\mathbb{P}$ on $\mathcal{X}_{V(\mathcal{G})}$ that is strictly positive, the following theorems are relevant:

Theorem 4 (Equivalence of Markov properties (Lauritzen, 1996)) Let $G = (V, E)$ be an UG with associated rvs $(X_v : v \in V)$ and $\mathbb{P}$ be a pmf on $\mathcal{X}_V$ that is strictly positive. Then:

$$\mathcal{I}_p(\mathcal{G}) \subseteq \mathcal{I}(\mathbb{P}) \Leftrightarrow \mathcal{I}_l(\mathcal{G}) \subseteq \mathcal{I}(\mathbb{P}) \Leftrightarrow \mathcal{I}_g(\mathcal{G}) \subseteq \mathcal{I}(\mathbb{P})$$

or in other words, $\mathcal{I}_p(\mathcal{G})$, $\mathcal{I}_l(\mathcal{G})$, or $\mathcal{I}_g(\mathcal{G})$ being an I-map for $\mathcal{I}(\mathbb{P})$ implies that the other independency sets from Definition 7 are also I-maps of $\mathcal{I}(\mathbb{P})$.

Theorem 5 (Hammersley & Clifford (1971)) Let $G = (V, E)$ be an UG with associated rvs $(X_v : v \in V)$ and $\mathbb{P}$ be a pmf on $\mathcal{X}_V$ that is strictly positive. Then the 3 Markov properties in Definition 7 are equivalent,

$$\mathcal{I}_p(\mathcal{G}) \subseteq \mathcal{I}(\mathbb{P}) \Leftrightarrow \mathcal{I}_l(\mathcal{G}) \subseteq \mathcal{I}(\mathbb{P}) \Leftrightarrow \mathcal{I}_g(\mathcal{G}) \subseteq \mathcal{I}(\mathbb{P})$$

and $\mathbb{P}$ is "Markovian", i.e. satisfies the Markov properties, if and only if it is a Gibbs distribution of the form,

$$\mathbb{P}(\boldsymbol{X} = \boldsymbol{x}) = \frac{1}{Z} \prod_{C \in \boldsymbol{C}(\mathcal{G})} \psi_C(\boldsymbol{x}_C)$$

where Z is a normalisation constant, $Z = \sum_{x \in \mathcal{X}} \prod_{C \in \boldsymbol{C}(\mathcal{G})} \psi_C(\boldsymbol{x}_C)$ and ψ_C are positive functions over the domain of the clique, $\psi_C : \mathcal{X}_C \to \mathbb{R}_+$

Due to its positivity, ψ_C can be written as an exponential

$$\psi_C(\boldsymbol{x}_C) = \exp\{\langle \boldsymbol{\theta}_C, \phi_C(\boldsymbol{x}_C) \rangle\}$$

where ϕ_C are sufficient statistics

$$\phi_C : \mathcal{X}_C \to \mathbb{R}^{|\mathcal{X}_C|}$$

The overcomplete sufficient statistic of discrete data is a "one-hot" vector that selects a specific weight value, e.g., $\psi_C(\boldsymbol{x}_C) = \exp\{\boldsymbol{\theta}_{C=\boldsymbol{x}_C}\}$. The full joint can be written in the famous exponential family form

$$\mathbb{P}(\boldsymbol{X} = \boldsymbol{x}) = \exp\{\langle \boldsymbol{\theta}, \phi(\boldsymbol{x})\rangle - \log Z)\rangle\}$$

where

$$\boldsymbol{\theta} = (\boldsymbol{\theta}_C \,:\, C \in \boldsymbol{C})$$
$$\phi(\boldsymbol{x}) = (\phi_C(\boldsymbol{x}_C) \,:\, C \in \boldsymbol{C})$$

The parameters of exponential family members can be estimated by minimising the negative average log-likelihood

$$\ell(\boldsymbol{\theta}; \mathcal{D}) = -\frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{x} \in \mathcal{D}} \log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x})$$

for some dataset $\mathcal{D}$, e.g. via first-order numeric optimisation methods. $\mathcal{D}$ contains samples from $\boldsymbol{X}$, and it can be shown that the estimated probability mass converges to the data generating distribution as the size of $\mathcal{D}$ increases (Koller & Friedman, 2009).

However, computing Z and hence performing probabilistic inference is #P-hard (Valiant, 1979; Bulatov & Grohe, 2004). Exact inference can be carried out via the junction tree algorithm (JTA), but requires a chordal graphical structure that is an I-map of $\mathbb{P}$, i.e. a graph structure relative to which $\mathbb{P}$ is decomposable (Pearl, 1988, p. 113).

2.2.3 Decomposable Models and the Conversion between Bayesian Networks and Markov Networks

A *decomposable model (DM)*, $\mathbb{P}_{\mathcal{G}}$, is a MN where the underlying conditional independence structure, $\mathcal{G}$, is a chordal graph, i.e. decomposable. In fact, any distribution $\mathbb{P}$ is decomposable relative to a chordal graph $\mathcal{G}$ as long as the conditional independencies in $\mathcal{G}$ is an I-map of $\mathbb{P}$ (Pearl, 1988, p. 115). Therefore, it is always possible to express a MN with a non-chordal graph structure by triangulating the MN's graph structure since adding edges to a UG will only reduce the number of conditional independencies encoded within the graph. This implies that the triangulated graph will be an I-map of the original, non-chordal graph, of the MN, and therefore also an I-map of the MN's distribution.

Now that we know we can convert any MN into a DM, the question is how we can convert a BN into a DM. The first logical step to take when converting BNs to DMs is to convert the DAG graph structure of the given BN into an UG.

[6] Adding edges between parents with a common child is also known as "marrying the parents". In fact the term "moral" stems from the notion that "marrying" parents of a common child is somehow "moral".

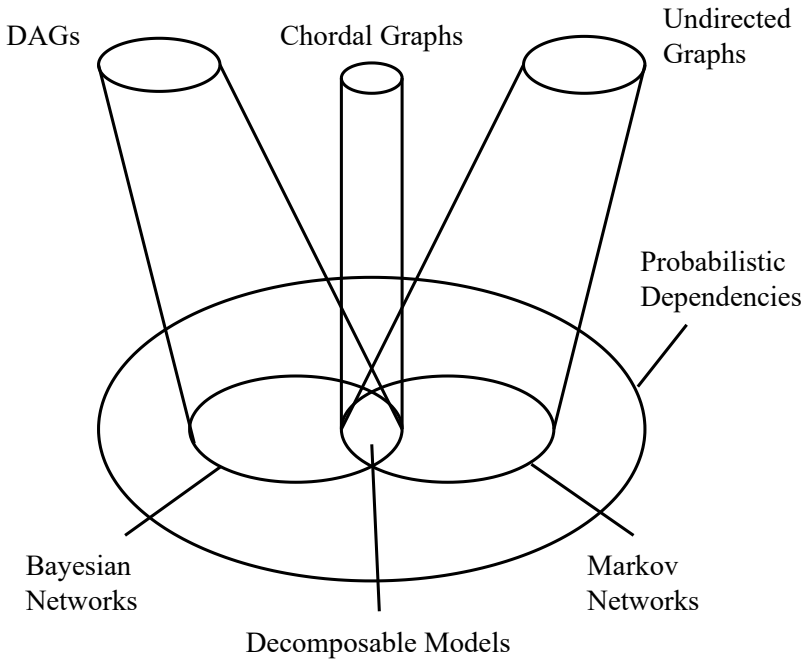


FIGURE 2.4: Relationship between probabilistic models and their graphical representations. Figure adapted from (Pearl, 1988, Figure 3.12).

Definition 8 (Moral graph $M[\vec{\mathcal{G}}]$ (Cowell et al. 1999, Section 3.2.1)) Let $\vec{\mathcal{G}}$ be a DAG. Then the *moral graph* of $\vec{\mathcal{G}}$, $M[\vec{\mathcal{G}}]$, is obtained by removing the directionality of all edges and adding additional undirected edges[6] between any vertices with a common child.

Furtheremore, for a BN $\mathcal{B} = (\vec{\mathcal{G}}, \mathbb{P})$, the moral graph of $\vec{\mathcal{G}}$ is also an I-map for $\mathbb{P}$ (Koller & Friedman, 2009, Corollary 4.2). We can then construct a new MN by using $M[\vec{\mathcal{G}}]$ as the graphical structure and the CPTs of $\mathcal{B}$ as the factors of this new MN (Koller & Friedman, 2009). Of course, the moral graph $M[\vec{\mathcal{G}}]$ might not be chordal itself (Cowell et al. 1999, p. 50), therefore, further triangulation might be needed to convert this new found MN into a DM.

It is also possible to convert a DM into either a MN or BN. The fact that we can convert a DM into a MN is trivally true as DMs are just MNs with a chordal graph structure. Converting DMs to BNs on the other hand involves the fact that for a chordal graph $\mathcal{G}$, the edges of $\mathcal{G}$ can be directed so that every pair of converging arrows emanates from two adjacent vertices, forming a DAG (Pearl, 1988, p. 127). We can then conclude that the class of probabilistic graphical models whose dependencies can be represented by both a DAG and an UG is the DM. The relationships between these probabilistic models and their respective graphical representations is illustrated in Figure 2.4.

One benefit of DMs, and the main property we will exploit throughout this thesis, is that it decomposes the joint distribution of a DM, $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$, into a product and quotient of probabilities over cliques in its graph structure $\mathcal{G}$.

Theorem 6 (Decomposing joint probability (Pearl, 1988, Theorem 8)) If $\mathbb{P}$ is decomposable relative to chordal graph $\mathcal{G}$, then the joint distribution of $\mathbb{P}$ can be decomposed as such:

$$\mathbb{P}_{\mathcal{G}}(\boldsymbol{x}) = \frac{\prod_{C \in \boldsymbol{C}} \mathbb{P}_C(\boldsymbol{x})}{\prod_{S \in \boldsymbol{S}} \mathbb{P}_S(\boldsymbol{x})}$$

where $\mathbb{P}_d(\cdot)$ represents the marginal probability over the domain $\mathcal{X}_d$.

Alternatively, we can also represent the joint distribution of $\mathbb{P}_{\mathcal{G}}$ as a product of conditional probability tables if we choose a random maximal clique in $\boldsymbol{C}$ to be the root node of $\mathbb{P}_{\mathcal{G}}$'s junction tree $\mathcal{T}$.

$$\mathbb{P}_{\mathcal{G}}(\boldsymbol{x}) = \prod_{C \in \boldsymbol{C}} \mathbb{P}_{C-\text{pa}(C)|\text{pa}(C)}(\boldsymbol{x}_{C-\text{pa}(C)}|\boldsymbol{x}_{\text{pa}(C)}) = \prod_{C \in \boldsymbol{C}} \mathbb{P}^{\mathcal{T}}_C(\boldsymbol{x})$$

where

$$\mathbb{P}^{\mathcal{T}}_C(\boldsymbol{x}) = \mathbb{P}_{C-\text{pa}(C)|\text{pa}(C)}(\boldsymbol{x}_{C-\text{pa}(C)}|\boldsymbol{x}_{\text{pa}(C)})$$

and $\text{pa}(C)$ is the parent clique of C in the junction tree $\mathcal{T}$. $\text{pa}(C) = \emptyset$ when C is the root node of $\mathcal{T}$.

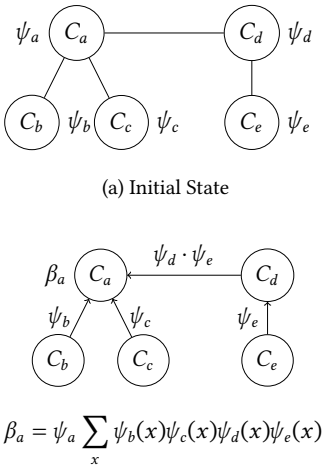
2.2.4 Junction Tree Algorithm

Recall the partition function of a general Markov network Z:

$$Z = \sum_{\boldsymbol{x} \in \mathcal{X}} \prod_{C \in \boldsymbol{C}} \psi_C(\boldsymbol{x}_C)$$

Evaluating the partition function of loopy models, i.e. models with a non-chordal graph structure, exactly does not necessarily require a naive summation over the state space $\mathcal{X}$; there is another, more efficient, technique. Any loopy graph can be converted into a tree, the so-called *junction tree* (JT) (Lauritzen & Spiegelhalter, 1988; Wainwright & Jordan, 2008; Koller & Friedman, 2009). In fact, as illustrated in Figure 2.5, the *junction tree algorithm* (JTA) is a belief propagation algorithm that goes a step further and computes the un-normalized marginal "probability", β, for each maximal clique in the JT (Koller & Friedman, 2009, Corollary 10.2):

$$\forall C \in \boldsymbol{C} \; : \; \beta_C(\boldsymbol{x}_C) = \sum_{\boldsymbol{x} \in \mathcal{X}_{X-C}} \prod_{C \in \boldsymbol{C}} \psi_C(\boldsymbol{x}, \boldsymbol{x}_C) \tag{2.2}$$



(a) Initial State

(b) After junction tree algorithm

Figure 2.5: Illustrations of the Junction Tree Algorithm.

As with belief propagation in ordinary trees, inference on the junction tree has a time complexity that is polynomial in the maximal state space size of its vertices. The maximal vertex state space size of a junction tree is, however, exponential in the size of the largest clique of a triangulation of $\mathcal{G}$, a.k.a. exponential in the treewidth of $\mathcal{G}$. Hence, if the treewidth of a loopy model after triangulation is small, exact inference via the junction tree algorithm is rather efficient.

Definition 9 (Treewitdh of a graph $\mathcal{G}$, $\omega(\mathcal{G})$) The treewith of any graph $\mathcal{G}$ is the number of vertices minus 1 in the largest maximal clique of the minimal triangulation of $\mathcal{G}$ (Koller & Friedman, 2009, Definition 9.6).

2.3 Entropy

For a discrete random variable/vector $\boldsymbol{X}$ with joint probability distribution $\mathbb{P}$, Shannon (1948) originally defined entropy as a measure of the uncertainty in the outcome of the random vector $\boldsymbol{X}$ using the following formula:

$$H(\mathbb{P}) = \sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{P}(\boldsymbol{x}) \log \frac{1}{\mathbb{P}(\boldsymbol{x})} \tag{2.3}$$

Therefore, Equation (2.3) is also known as the Shannon entropy. Rényi (1961) then proposed the following generalisation of the Shannon entropy

$$H_\alpha(\mathbb{P}) = \sum_{\boldsymbol{x} \in \mathcal{X}} \frac{1}{1-\alpha} \log \sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{P}(\boldsymbol{x})^\alpha \tag{2.4}$$

where α is a parameter of the entropy functional. This generalisation then converges to the Shannon entropy when the parameter α approches 1 (Rényi, 1961):

$$H_1(\mathbb{P}) := \lim_{\alpha \to 1} H_\alpha(\mathbb{P}) = \sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{P}(\boldsymbol{x}) \log \frac{1}{\mathbb{P}(\boldsymbol{x})} \tag{2.5}$$

The generalisation in Equation (2.4) would later be known as the Rényi entropy or the α-entropy (Csiszar, 1995).

2.3.1 Conditional Entropy

Assume we wish to find the entropy of the random vector $\boldsymbol{Y}$ given the random vector $\boldsymbol{Z}$ under the conditional distribution $\mathbb{P}_{Y|Z}$. Note the difficulty in computing the conditional entropy, as opposed to the normal joint entropy, is that $\mathbb{P}_{Y|Z}$ actually constitutes multiple probability distributions over $\boldsymbol{Y}$, $\mathbb{P}_{Y|z}$, one for each possible value of $\boldsymbol{z} \in \mathcal{Z}$.

Since $\mathbb{P}_{Y|z}$ are normal probability distributions over Y, we do know how to find the Shannon entropy for them

$$\forall z \in \mathcal{Z} \; : \; H(\mathbb{P}_{Y|z}) = -\sum_{y \in \mathcal{Y}} \mathbb{P}_{Y|z}(y) \log \mathbb{P}_{Y|z}(y) \tag{2.6}$$

Then, a natural way to define the conditional *Shannon* entropy is to take the expectation of $H(\mathbb{P}_{Y|Z})$ with respect to $\mathbb{P}_Z$ (Cover & Thomas, 2006, p. 17):

$$H(\mathbb{P}_{Y|Z}) = \mathbb{E}_{z \sim \mathbb{P}_z}[H(\mathbb{P}_{Y|Z=z})] = -\sum_{z \in \mathcal{Z}} \mathbb{P}_Z(z) \sum_{y \in \mathcal{Y}} \mathbb{P}_{Y|z}(y) \log \mathbb{P}_{Y|z}(y) \tag{2.7}$$

The "naturalness" of this definition for the conditional Shannon entropy is demonstrated by the fact that this definition gives the chain rule of Shannon entropy (Cover & Thomas, 2006, Theorem 2.2.1):

$$H(\mathbb{P}_{Y,Z}) = H(\mathbb{P}_Z) + H(\mathbb{P}_{Y|Z}) \tag{2.8}$$

However, this relation with the expected conditional divergence in Equation (2.7) and the chain rule in Equation (2.8) only exists for the Shannon entropy. Therefore, the question of obtaining a "natural" definition of the Rényi entropy is more complex as evidenced by the numerous proposals of varying definitions for a conditional Rényi entropy in the literature (Teixeira et al. 2012; Fehr & Berens, 2014; Ilić et al. 2017).

2.4 Divergence

A divergence is a measure of the "difference" between 2 probability distributions. More formally, a divergence is a functional between 2 distributions that satisfies certain properties as defined in Definition 10.

Definition 10 (Divergence) Suppose P is the set of probability distributions with the same support. A divergence, D, is the function[7]:

$$D(\cdot\|\cdot) \; : \; P \times P \to \mathbb{R}$$

such that $\forall \mathbb{P}, \mathbb{Q} \in P$:

$$D(\mathbb{P}\|\mathbb{Q}) \geq 0$$
$$\mathbb{P} = \mathbb{Q} \Leftrightarrow D(\mathbb{P}\|\mathbb{Q}) = 0$$

Examples of popular divergences include the Kullback-Leibler divergence (Kullback & Leibler, 1951), the Hellinger distance (Hellinger, 1909), and the χ^2 divergence (Pearson, 1900).

[7] Some authors also require that the quadratic part of the Taylor expansion of $D(p, p + dp)$ define a Riemannian metric on P (Amari, 2016). However, this requirement is not needed by the methods described in this thesis. Therefore, we will leave this requirement out of this definition.

Furthermore, there are also generalized divergences where common divergences, such as the ones mentioned above, are special cases of the generalized divergence. Specifically, we will use the generalized divergence known as the $\alpha\beta$-divergence (Cichocki et al. 2011).

Definition 11 ($\alpha\beta$-divergence) The $\alpha\beta$-divergence, D_{AB}, between 2 positive measures $\mathbb{P}$ and $\mathbb{Q}$ is defined by the following, where α and β are parameters:

$$D_{AB}^{(\alpha,\beta)}(\mathbb{P},\mathbb{Q}) = -\frac{1}{\alpha\beta}\sum_{x\in\mathcal{X}}\left(\mathbb{P}(x)^{\alpha}\mathbb{Q}(x)^{\beta} - \frac{\alpha}{\alpha+\beta}\mathbb{P}(x)^{\alpha+\beta} - \frac{\beta}{\alpha+\beta}\mathbb{Q}(x)^{\alpha+\beta}\right) \quad \text{for } \alpha,\beta,\alpha+\beta\neq 0 \tag{2.9}$$

To avoid indeterminacy or singularity for certain values of α, β in Equation (2.9), we can extend the $\alpha\beta$-divergence by continuity, using l'Hôpital's formula, to cover values of $\alpha, \beta \in \mathbb{R}$:

$$D_{AB}^{\alpha,\beta}(\mathbb{P},\mathbb{Q}) = \sum_{x\in\mathcal{X}} d_{AB}^{\alpha,\beta}\big(\mathbb{P}(x),\mathbb{Q}(x)\big)$$

where

$$d_{AB}^{(\alpha,\beta)}(\mathbb{P}(x),\mathbb{Q}(x)) = \begin{cases} -\frac{1}{\alpha\beta}\left(\mathbb{P}(x)^{\alpha}\mathbb{Q}(x)^{\beta} - \frac{\alpha\mathbb{P}(x)^{\alpha+\beta}}{\alpha+\beta} - \frac{\beta\mathbb{Q}(x)^{\alpha+\beta}}{\alpha+\beta}\right) & \text{for } \alpha,\beta,\alpha+\beta\neq 0 \\ \frac{1}{\alpha^2}\left(\mathbb{P}(x)^{\alpha}\log\frac{\mathbb{P}(x)^{\alpha}}{\mathbb{Q}(x)^{\alpha}} - \mathbb{P}(x)^{\alpha} + \mathbb{Q}(x)^{\alpha}\right) & \text{for } \alpha\neq 0,\beta=0 \\ \frac{1}{\alpha^2}\left(\log\frac{\mathbb{Q}(x)^{\alpha}}{\mathbb{P}(x)^{\alpha}} + \left(\frac{\mathbb{Q}(x)^{\alpha}}{\mathbb{P}(x)^{\alpha}}\right)^{-1} - 1\right) & \text{for } \alpha=-\beta\neq 0 \\ \frac{1}{\beta^2}\left(\mathbb{Q}(x)^{\beta}\log\frac{\mathbb{Q}(x)^{\beta}}{\mathbb{P}(x)^{\beta}} - \mathbb{Q}(x)^{\beta} + \mathbb{P}(x)^{\beta}\right) & \text{for } \alpha=0,\beta\neq 0 \\ \frac{1}{2}(\log\mathbb{P}(x) - \log\mathbb{Q}(x))^2 & \text{for } \alpha,\beta=0. \end{cases} \tag{2.10}$$

The parameters α and β in the $\alpha\beta$-divergence are used to express other commonly used divergences. Specifically, the parameters $\alpha = 1, \beta = 0$ give the Kullback-Leibler divergence, while the parameters $\alpha = 0.5, \beta = 0.5$ give the Bhattacharyya coefficient which immediately gives the Hellinger distance (Cichocki et al. 2011).

2.4.1 Conditional Divergence

Similar to the conditional entropy in Section 2.3.1, the difficulty in measuring the divergence between 2 conditional distributions, $\mathbb{P}_{Y|Z}$ and $\mathbb{Q}_{Y|Z}$, is that they are made up of multiple probability distributions over Y, $\mathbb{P}_{Y|Z=z}$ and $\mathbb{Q}_{Y|Z=z}$, one for each value of $z \in \mathcal{Z}$.

However, due to the relationship between the KL-divergence and Shannon entropy, there is a "natural" definition for the *conditional* KL-divergence resulting from the fact that the following 2 natural approaches to define the conditional KL-divergence lead to the same result (Bleuler et al. 2020, eq. 2, 3)

$$D(\mathbb{P}_{Y|Z} \parallel \mathbb{Q}_{Y|Z} \mid \mathbb{P}_Z) := \sum_{z \in \mathcal{Z}} \mathbb{P}_Z(z) D(\mathbb{P}_{Y|Z=z} \parallel \mathbb{Q}_{Y|Z=z}) \tag{2.11}$$

$$= D(\mathbb{P}_{Y|Z}\mathbb{P}_Z \parallel \mathbb{Q}_{Y|Z}\mathbb{P}_Z) \tag{2.12}$$

where we are measuring the conditional divergence between $\mathbb{P}_{Y|Z}$ and $\mathbb{Q}_{Y|Z}$ in relation to the "base" distribution $\mathbb{P}$.

Unfortunately, similar to conditional entropy, this equivalence between the definitions of a conditional distribution in Equations (2.11) and (2.12) is unique to the KL-divergence. When proposing a possible definition of a conditional divergence for other divergences, previous approaches have either exclusively taken the definition in Equation (2.11) (Csiszar, 1995; Poczos & Schneider, 2012; Bhattacharyya & Chakraborty, 2018) or Equation (2.12) (Sibson, 1969; Cai & Verdú, 2019; Bleuler et al. 2020) as the basis for their definition.

Therefore, in this thesis we will use the definition in Equation (2.11) as the basis of our definition of a conditional $\alpha\beta$-divergence. This definition involves taking the expectation of $D(\mathbb{P}_{Y|Z=z}, \mathbb{Q}_{Y|Z=z})$ with respect to $\mathbb{P}_Z$.

Definition 12 (General conditional divergence)

$$D(\mathbb{P}_{Y|Z} \parallel \mathbb{Q}_{Y|Z}) = \mathbb{E}_{Z \sim \mathbb{P}} \left[D(\mathbb{P}_{Y|Z} \parallel \mathbb{Q}_{Y|Z}) \right]$$
$$= \sum_{z \in \mathcal{Z}} \mathbb{P}_Z(z) D(\mathbb{P}_{Y|Z=z} \parallel \mathbb{Q}_{Y|Z=z})$$

2.5 Concept Drift

Due to the fact that the world is a dynamic system, it is not rare to have the processes underlying some data stream to change causing the resulting data to express different "concepts" over time. These changes in "concepts", also known as *concept drift*, can cause the performance of models and classifiers trained on an initial set of data to deteriorate over time.

Formally, concept drift can be defined as changes to the joint distribution $\mathbb{P}(y, X)$, also known as Joint Drift, where X is the set of covariate variables and y is the class variable (Gama et al. 2014). Furthermore, it might be relevant to also study the different components of the joint distribution:

- **Covariate Drift** Changes to the covariate distribution, $\mathbb{P}(X)$
- **Posterior Drift** Changes to the distribution of the class given the covariate value, $\mathbb{P}(y|X)$

The are many ways to characterise occurrences of concept drift, both qualitatively and quantitatively, between datasets obtained from differing contexts, or within a data stream over time (Webb et al. 2016). However, this project will only focus on the 2 of these characteristics, *drift duration* and *drift magnitude*.

Drift duration is a quantitative measure on the size of the transition period between 2 concepts (Webb et al. 2016). This measure of drift unifies 2 existing qualitative characteristics of drift in the literature, namely abrupt and gradual drift (Tsymbal, 2004).

Another useful quantitative measure of concept drift is a measure of the degree of difference between the different concepts within a data stream or across different datasets. This quantitative measure is also known in the literature as *drift magnitude* (Webb et al. 2016). Since this measure of drift is relatively new, there has not been much research done into determining the best method to use when measuring the distance between concepts. Previous approaches to calculating *drift magnitude* have used statistical distance metrics such as the Total Variation distance (Webb et al. 2018). However, this thesis will use the $\alpha\beta$-divergence as described in Section 2.4.

When developing new learners and techniques to adapt to occurrences of concept drift, it is vital to have available datasets that are known to contain occurrences of concept drift. We require these datasets not only to test and compare the technique being developed with previous techniques, but also to understand the characteristics and response the technique will have on varying characteristics of concept drift. Therefore, the creation of datasets with known occurrences of concept drift is

The problem of changing distributions is one that occurs throughout the fields of machine learning, and therefore manifests slightly differently, with differing names, based on the problem being tackled in each field. For instance, there has been a lot of work in out-of-distribution, or domain, generalisation in recent years (Wang et al. 2022; Zhou et al. 2022). Domain generalisation involves learning a model from multiple different, but related, domains that is capable of generalising to unseen domains. Similar to concept drift, these domains can have wildly different distributions. However, unlike in concept drift, quite a lot of work in domain generalisation assumes a relatively stable $\mathbb{P}(y|X)$ distribution (Zhou et al. 2022). Furthermore, most of the work in domain generalisation focuses on non-tabular data, while most of the work in concept drift focuses on tabular data.

2.6 Datasets with Concept Drift

In this section, we will review and discuss some existing methods to generate data containing concept drift, which we will hereby refer to as drift generators. The main purpose of this discussion will be to review and extract any insights that might be useful developing our own high-dimensional drift generator.

We will divide these drift data generators into 2 categories: generators of concept drift with unknown magnitudes, and generators of concept drift with known magnitudes. Historically, only generators of the former category have been used. However, recently there has been an increase in the use of generators in the latter category. Ultimately, generators of concept drift with known magnitudes are desirable as they allow us to control for the magnitude of concept drift in any experiments carried out on the generated datasets. This allows us to better understand the behaviour of the model or adaptation technique being tested on varying conditions, and in the end, will provide better insight into what situations a certain model or adaptation technique performs best at.

2.6.1 Unknown Drift Magnitudes

2.6.1.1 Synthetic Drift Generators

One of the first approaches into obtaining datasets containing concept drift was to create generators that generated data based on some predefined problem or model. Concept drift is then created by changing the problem or model used to generate the data during the data generation process. A common issue that synthetic drift generators share is that the relationships between their attributes, or inter-attribute structure, might be a bit simplistic and not representative of inter-attribute structures found in real-world data. For the sake of completeness, we will provide a brief overview of popular synthetic drift generators and their origins.

One of the first algorithms to track concept drift in a dataset was STAGGER (Schlimmer & Granger, 1986). STAGGER characterized a concept as a single Boolean predicate over the attributes in a dataset. The class of any instance in this dataset is then determined by the Boolean value returned by the predicate when the instance is passed as input. Different concepts are then defined by different predicates over the attributes. This definition of concept was later use by (Widmer & Kubat, 1996) in order to create a dataset to test their method FLORA2 against STAGGER.

The Streaming Ensemble Algorithm (SEA) is an ensemble classification algorithm capable of adapting to drift (Street & Kim, 2001). In order to test SEA, the authors

created a synthetic dataset with 3 attributes: f_1, f_2, and f_3. The class of any instance is then determined by the following Boolean function: $f_1 + f_2 <= \theta$, where different values of θ define different concepts.

Similar to (Street & Kim, 2001), (Minku et al. 2010) used mathematical functions to define their classification problems in their test datasets. Minku et al. created 2 synthetic datasets. The first dataset determined the class of any instance, (x, y), by determining if they are inside or outside the ellipse $(x - a)^2 + (y - b)^2 = r^2$. The second dataset determined the class of each instance, (x, y), by determining if they are above or below the sinusoidal wave $y = a \sin(bx + c) + d$. For both these datasets, different concepts can be created by changing the parameters a, b, c, and d.

2.6.1.2 Manipulating Real-World Datasets

Another approach to creating datasets that contain concept drift is to manipulate some real world dataset. The main benefit of this method over synthetic drift generators is that the resulting datasets will have more realistic inter-attribute structures.

This approach to generating datasets with distributional shifts is quite common in the training and testing of image classifiers. For instance, one could rotate the images in a dataset to produce images with a different covariate distribution but with the same class (Ghifary et al. 2015). In general, manipulating and applying transformations to audio-visual data can be done based on human intuition on which transformation is realistic and will be useful for the training and testing of models.

However, such intuitions might not exist for tabular data. Therefore we will give an overview of the approaches explored by previous research to manipulate tabular datasets as these methods might prove useful in developing our own high-dimensional drift generator.

Merging Datasets. One method to create a dataset containing concept drift from real-world datasets is to "merge" 2 real world datasets together (Bifet et al. 2009a). Let $\mathcal{D}_x$ and $\mathcal{D}_y$ be the datasets we want to merge with the variables (X_1, X_2, C_x) and (Y_1, C_y) respectively, where C_x and C_y are the class variable of these datasets. In order to merge $\mathcal{D}_x$ and $\mathcal{D}_y$, a new dataset, $\mathcal{D}_z$, is created that contains the non-class variables of both $\mathcal{D}_x$ and $\mathcal{D}_y$. Instead of having both C_x and C_y in $\mathcal{D}_z$, a new class variable, C_z, is introduced that can take all the values of both C_x and C_y. Therefore, the new dataset, $\mathcal{D}_z$, will have the variables (X_1, X_2, Y_1, C_z). The i-th instance in $\mathcal{D}_z$ is then defined as the concatenation of the i-th instance in $\mathcal{D}_x$ and $\mathcal{D}_y$ without

their classes. The class of the i-th instance in $\mathcal{D}_z$, $C_{z,i}$, is then determined based on whether the i-th instance occurs before or after concept drift. If it is before drift, then $C_{z,i} = C_{x,i}$, otherwise $C_{z,i} = C_{y,i}$.

Attribute Value Transformation. Another method to manipulate some dataset to contain concept drift would be to transform the values of an attribute to another value after concept drift. Specifically, Klinkenberg (2001) effectively did such a transformation on the class attribute on the real-world dataset they used. They had a dataset of business news text that were categorised into 5 categories. These categories were effectively the classes of the documents. They then divided these documents into 20 batches and added a new variable, the probability that the category is relevant to the reader. The probability that each category is relevant to the reader is then varied across the different batches to simulate concept drift between the different batches.

Vary Value Frequency. Another manipulation method would be to partition the original dataset such that the frequency of the values of some attributes are different over the different partitions (Robert Polikar et al. 2001). This would cause drift to occur due to the difference in the probability distribution over these attributes.

2.6.2 Known Drift Magnitude

As we will see below, a common characteristic of these methods is that they first take some generative model of a probability distribution, ϕ. Note that ϕ can be synthetically generated or learnt on some existing data. They then find a ϕ^*, such that the distance between the distribution of ϕ and ϕ^* is some ε away from a given magnitude, according to some chosen divergence. A dataset with concept drift can then be generated by sampling ϕ before concept drift and sampling ϕ^* after drift. Our proposed high-dimensional drift generator will follow this rough procedure as well.

2.6.2.1 Naive Bayes Model

Webb et al. (2016) created a synthetic drift generator capable of generating categorical datasets with concept drift of a given magnitude. This was achieved by creating a Bayesian network with 5 covariate attributes and 1 class attribute, where each covariate attribute is a parent of the class attribute. The multinomial probability of each parent attribute is sampled using a flat Dirichlet with concentration parameter 1. A random class value is assigned for each combination of values over the parent attributes. The initial model is then used to generate instances before the drift.

This generator can also generate either pure class or pure covariate drift of a given magnitude calculated using Hellinger distance. To generate pure class drift, the generator changes the class for k covariate value combinations, where k can be calculated based on the given target drift magnitude. To generate pure covariate drift, the generator just samples different parameters for the parents until the target drift magnitude is achieved. This new modified model is then used to generate instances after drift.

The main issue of this generator is that it is incapable of representing more complex interactions between the covariate attributes and assumes the existence of a class variable of interest. Therefore, it is not very useful for creating datasets where the main source of distributional changes are primarily from the covariate attributes.

2.6.2.2 Controlling Change Magnitude (CCM)

Although this thesis focuses on discrete distributions, we shall make a brief mention of a method by Carrera & Boracchi (2018), called Controlling Change Magnitude (CCM), that is capable of generating high-dimensional *numeric* datasets with known drift magnitudes. CCM first fits a Gaussian Mixture (GM) model, ϕ, on some data. CCM then uses an iterative algorithm to find a second GM model ϕ^* that corresponds to a roto-translation of the original data that is some distance from the original dataset. During this process, the distance between 2 GMs is calculated using the symmetric Kullback-Leibler divergence, $D_{KL}(\mathbb{P}||\mathbb{Q}) + D_{KL}(\mathbb{Q}||\mathbb{P})$, also known as Jeffreys divergence (Jeffreys, 1998).

Unlike the generator in Section 2.6.2.1, this generator is capable of generating concept drift in high dimensions. However, this method is incapable of generating concept drift in categorical data. Therefore, our proposed high-dimensional drift generator will attempt to fill this gap in the literature.

Part I

Computing Divergences between Graphical Models

Chapter 3

Computing Divergences between the Joint Distribution of Decomposable Models

Recall from Section 1.2 that one of the main problems we wish to solve is the computation of divergences between high-dimensional discrete distributions. Furthermore, we also proposed using probabilistic graphical models in order to approach this problem using more compact and efficient representations of discrete distributions. This approach has been used in the past by Moral et al. (2021) where they proposed a method to compute the KL divergence between 2 BNs. However, their method is limited to computing the KL divergence, and in the context of measuring the magnitude of concept drift, we will frequently encounter situations where the KL divergence is undefined, i.e. situations where $\mathbb{P}(\boldsymbol{x}) > 0$ but $\mathbb{Q}(\boldsymbol{x}) = 0$.

Motivated by these considerations, in this chapter we show how to efficiently compute a wide family of divergences, the $\alpha\beta$-divergence, between the joint distribution of two DMs. This will allow us to use divergences other than the KL divergence, such as the Hellinger distance. In the process of showing how the $\alpha\beta$-divergence can be computed between any two DMs, we will reach a more general result. That is, we will show how one can compute, between the two joint distributions of DMs, the functional $\mathcal{F}$ defined in Definition 13:

Definition 13 (Functional $\mathcal{F}$)

$$\mathcal{F}(\mathbb{P}, \mathbb{Q}; g, h, g^*, h^*) = \sum_{x \in \mathcal{X}} \Big(g\big[\mathbb{P}\big](\boldsymbol{x})\Big)\Big(h\big[\mathbb{Q}\big](\boldsymbol{x})\Big) L\Big(\Big(g^*[\mathbb{P}](\boldsymbol{x})\Big)\Big(h^*[\mathbb{Q}](\boldsymbol{x})\Big)\Big)$$

where, L is any function with the property:

$$L\left(\prod_x x\right) = \sum_x L(x) \tag{3.1}$$

and for any probability mass function P over variables $\boldsymbol{X}$, $f \in \{g, h, g^*, h^*\}$ are functionals with the property:

$$f\left[\prod_{Z \in \mathcal{A}} P_Z\right] = \prod_{Z \in \mathcal{A}} f\left[P_Z\right] \tag{3.2}$$

where

$$\mathcal{A} \subset \mathcal{P}(\boldsymbol{X}) \quad \text{s.t.} \quad \prod_{Z \in \mathcal{A}} P_Z = P_{\cup \mathcal{A}} \tag{3.3}$$

and $\bigcup \mathcal{A}$ is the set of all the variables in the sets of $\mathcal{A}$.

An example of a functional f with the property in Equation (3.2) is the basic power function $f[P](x) = P(x)^a$ for $a \in \mathbb{R}$. In general, any functional will satisfy the property in Equation (3.2) if it satisfies the property $f\left[\prod_{g \in G} g\right] = \prod_{g \in G} f[g]$, where G is some set of functions where the product between them is defined.

This result implies the possibility for the computation of further divergences and functionals between two DMs in addition to the $\alpha\beta$-divergence because the class of functionals that can be expressed as a linear combination of function $\mathcal{F}$ in Definition 13 is much wider than the class of $\alpha\beta$-divergence. Furthermore, as we will see later in this chapter, by focusing on computing the functional $\mathcal{F}$ between the joint distributions of 2 DMs, the exposition of this chapter will be more succinct as it unifies the different cases of the $\alpha\beta$-divergence into a linear combination of one, parameterised, functional.

This chapter is organised as follows. In Section 3.1, we discuss the $\alpha\beta$-divergence and its application in measuring the divergence between two DMs. In Section 3.2, we present our method for computing the sum-product over multiple chordal graphs and show in Section 3.3 how it can be used to compute the $\alpha\beta$-divergence between two DMs. In Section 3.4, we discuss the final complexity of computing the $\alpha\beta$-divergence between two DMs. In Section 3.5, we will compare the runtime of our method with the method by Moral et al. Then in Section 3.6, we will present a case study into why using divergences other than the KL divergence is useful in the context of model selection, a problem that the KL divergence is normally well suited to and widely used in. Finally, in Section 3.7 we close the chapter with some final remarks.

3.1 $\alpha\beta$-Divergence and Functional $\mathcal{F}$ between Joint Distributions of Decomposable Models

In this section, we will revisit the $\alpha\beta$-divergence from Definition 11, and show how, in cases when α and β are not both zero, we can express it in terms of parameterisations of the functional $\mathcal{F}$ from Definition 13. We then show that, when the $\alpha\beta$-divergence is not expressible in terms of $\mathcal{F}$ functionals, i.e. when $\alpha, \beta = 0$, the $\alpha\beta$-divergence between the joint distributions of two DMs can be computed directly efficiently. Therefore, computing the $\alpha\beta$-divergence between the joint distributions of DMs hinges on computing $\mathcal{F}$ between these distributions. As such, we end this section by plugging in the joint distributions of DMs into $\mathcal{F}$ and simplifying the expression to set up the problem that we will tackle in the next section, Section 3.2.

First, recall the $\alpha\beta$-divergence from Definition 11:

Definition 11 ($\alpha\beta$-divergence) The $\alpha\beta$-divergence, D_{AB}, between 2 positive measures $\mathbb{P}$ and $\mathbb{Q}$ is defined by the following, where α and β are parameters:

$$D_{AB}^{(\alpha,\beta)}(\mathbb{P},\mathbb{Q}) = -\frac{1}{\alpha\beta}\sum_{\boldsymbol{x}\in\mathcal{X}}\left(\mathbb{P}(\boldsymbol{x})^{\alpha}\mathbb{Q}(\boldsymbol{x})^{\beta} - \frac{\alpha}{\alpha+\beta}\mathbb{P}(\boldsymbol{x})^{\alpha+\beta} - \frac{\beta}{\alpha+\beta}\mathbb{Q}(\boldsymbol{x})^{\alpha+\beta}\right) \quad \text{for } \alpha,\beta,\alpha+\beta\neq 0 \tag{2.9}$$

To avoid indeterminacy or singularity for certain values of α, β in Equation (2.9), we can extend the $\alpha\beta$-divergence by continuity, using l'Hôpital's formula, to cover values of $\alpha, \beta \in \mathbb{R}$:

$$D_{AB}^{\alpha,\beta}(\mathbb{P},\mathbb{Q}) = \sum_{\boldsymbol{x}\in\mathcal{X}} d_{AB}^{\alpha,\beta}\big(\mathbb{P}(\boldsymbol{x}),\mathbb{Q}(\boldsymbol{x})\big)$$

where

$$d_{AB}^{(\alpha,\beta)}(\mathbb{P}(\boldsymbol{x}),\mathbb{Q}(\boldsymbol{x})) = \begin{cases} -\frac{1}{\alpha\beta}\left(\mathbb{P}(\boldsymbol{x})^{\alpha}\mathbb{Q}(\boldsymbol{x})^{\beta} - \frac{\alpha\mathbb{P}(\boldsymbol{x})^{\alpha+\beta}}{\alpha+\beta} - \frac{\beta\mathbb{Q}(\boldsymbol{x})^{\alpha+\beta}}{\alpha+\beta}\right) & \text{for } \alpha,\beta,\alpha+\beta\neq 0 \\ \frac{1}{\alpha^2}\left(\mathbb{P}(\boldsymbol{x})^{\alpha}\log\frac{\mathbb{P}(\boldsymbol{x})^{\alpha}}{\mathbb{Q}(\boldsymbol{x})^{\alpha}} - \mathbb{P}(\boldsymbol{x})^{\alpha} + \mathbb{Q}(\boldsymbol{x})^{\alpha}\right) & \text{for } \alpha\neq 0,\beta=0 \\ \frac{1}{\alpha^2}\left(\log\frac{\mathbb{Q}(\boldsymbol{x})^{\alpha}}{\mathbb{P}(\boldsymbol{x})^{\alpha}} + \left(\frac{\mathbb{Q}(\boldsymbol{x})^{\alpha}}{\mathbb{P}(\boldsymbol{x})^{\alpha}}\right)^{-1} - 1\right) & \text{for } \alpha=-\beta\neq 0 \\ \frac{1}{\beta^2}\left(\mathbb{Q}(\boldsymbol{x})^{\beta}\log\frac{\mathbb{Q}(\boldsymbol{x})^{\beta}}{\mathbb{P}(\boldsymbol{x})^{\beta}} - \mathbb{Q}(\boldsymbol{x})^{\beta} + \mathbb{P}(\boldsymbol{x})^{\beta}\right) & \text{for } \alpha=0,\beta\neq 0 \\ \frac{1}{2}(\log\mathbb{P}(\boldsymbol{x}) - \log\mathbb{Q}(\boldsymbol{x}))^2 & \text{for } \alpha,\beta=0. \end{cases} \tag{2.10}$$

The full expression for the $\alpha\beta$-divergence in Equation (2.10) can be further simplified as a linear combination of 3 smaller functionals.

Theorem 7 (Expressing the $\alpha\beta$-divergence in terms of 3 functionals) The 5 cases of the $\alpha\beta$-divergence can be re-expressed into a linear combination of the following 3 functionals in linear time:

$$f_1(\mathbb{P},\mathbb{Q}) = \sum_{x\in\mathcal{X}} \frac{1}{2}(\log \mathbb{P}(x) - \log \mathbb{Q}(x))^2$$
$$f_2(\mathbb{P},\mathbb{Q};a,b) = \sum_{x\in\mathcal{X}} \mathbb{P}(x)^a \mathbb{Q}(x)^b$$
$$f_3(\mathbb{P},\mathbb{Q};a,b,c,d) = \sum_{x\in\mathcal{X}} \mathbb{P}(x)^a \mathbb{Q}(x)^b \log\left(\mathbb{P}(x)^c \mathbb{Q}(x)^d\right)$$

Proof When $\alpha,\beta,\alpha+\beta\neq 0$:

$$\begin{aligned} &D_{AB}^{(\alpha,\beta)}(\mathbb{P},\mathbb{Q}) \\ &= -\frac{1}{\alpha\beta}\left(\sum_{x\in\mathcal{X}} \mathbb{P}(x)^\alpha \mathbb{Q}(x)^\beta - \frac{\alpha}{\alpha+\beta}\sum_{x\in\mathcal{X}} \mathbb{P}(x)^{\alpha+\beta} - \frac{\beta}{\alpha+\beta}\sum_{x\in\mathcal{X}} \mathbb{Q}(x)^{\alpha+\beta}\right) \\ &= -\frac{1}{\alpha\beta}\left(f_2(\mathbb{P},\mathbb{Q};\alpha,\beta) - \frac{\alpha}{\alpha+\beta} f_2(\mathbb{P},\mathbb{Q};\alpha+\beta,0) - \frac{\beta}{\alpha+\beta} f_2(\mathbb{P},\mathbb{Q};0,\alpha+\beta)\right) \end{aligned}$$

When $\alpha\neq 0,\beta=0$:

$$\begin{aligned} D_{AB}^{(\alpha,0)}(\mathbb{P}\|\mathbb{Q}) &= \frac{1}{\alpha^2}\left(\sum_{x\in\mathcal{X}} \mathbb{P}(x)^\alpha \ln\frac{\mathbb{P}(x)^\alpha}{\mathbb{Q}(x)^\alpha} - \sum_{x\in\mathcal{X}} \mathbb{P}(x)^\alpha + \sum_{x\in\mathcal{X}} \mathbb{Q}(x)^\alpha\right) \\ &= \frac{1}{\alpha^2}\left(f_3(\mathbb{P},\mathbb{Q};\alpha,0,\alpha,-\alpha) - f_2(\mathbb{P},\mathbb{Q};\alpha,0) + f_2(\mathbb{P},\mathbb{Q};0,\alpha)\right) \end{aligned}$$

When $\alpha=-\beta\neq 0$:

$$\begin{aligned} D_{AB}^{(\alpha,-\alpha)}(\mathbb{P}\|\mathbb{Q}) &= \frac{1}{\alpha^2}\left(\sum_{x\in\mathcal{X}} \ln\frac{\mathbb{Q}(x)^\alpha}{\mathbb{P}(x)^\alpha} + \sum_{x\in\mathcal{X}}\left(\frac{\mathbb{Q}(x)^\alpha}{\mathbb{P}(x)^\alpha}\right)^{-1} - \sum_{x\in\mathcal{X}} 1\right) \\ &= \frac{1}{\alpha^2}\left(f_3(\mathbb{P},\mathbb{Q};0,0,-\alpha,\alpha) + f_3(\mathbb{P},\mathbb{Q};0,0,\alpha,-\alpha) - |\mathcal{X}|\right) \end{aligned}$$

When $\alpha=0,\beta\neq 0$:

$$\begin{aligned} D_{AB}^{(0,\beta)}(\mathbb{P}\|\mathbb{Q}) &= \frac{1}{\beta^2}\left(\sum_{x\in\mathcal{X}} \mathbb{Q}(x)^\beta \ln\frac{\mathbb{Q}(x)^\beta}{\mathbb{P}(x)^\beta} - \sum_{x\in\mathcal{X}} \mathbb{Q}(x)^\beta + \sum_{x\in\mathcal{X}} \mathbb{P}(x)^\beta\right) \\ &= \frac{1}{\beta^2}\left(f_3(\mathbb{P},\mathbb{Q};0,\beta,-\beta,\beta) - f_2(\mathbb{P},\mathbb{Q};0,\beta) + f_2(\mathbb{P},\mathbb{Q};\beta,0)\right) \end{aligned}$$

When $\alpha, \beta = 0$:

$$D_{AB}^{(0,0)}(\mathbb{P}\|\mathbb{Q}) = \frac{1}{2}(\sum_{x\in\mathcal{X}} \ln \mathbb{P}(\boldsymbol{x}) - \sum_{x\in\mathcal{X}} \ln \mathbb{Q}(\boldsymbol{x}))^2 = f_1(\mathbb{P}, \mathbb{Q})$$

Therefore, the ability to tractably compute these three functionals between 2 decomposable models will imply the ability to tractably compute the $\alpha\beta$-divergence between 2 decomposable models. Here, we assume a time complexity exponential to the treewidth of our decomposable models is an acceptable reduction in complexity compared to being exponential to the number of variables n for it to be *tractable*.

This assumption of tractability is true when the graphical structure representing the conditional independencies in the distribution is sparse and has a low treewidth. However, a recent experimental study on the treewidth of networks in the real world have shown that most real world networks have treewidths large enough to render a complexity exponential to the treewidth intractable (Maniu et al. 2019). Whether the same is true for conditional independence structures for distributions in the real world requires more research. Intuitively, the graph representing the conditional independencies between a set of variables is likely to be a subgraph of the network containing an edge between any variables related to each other, as potentially only a subset of edges in the latter network is needed to express the existing conditional independencies.

Theorem 8 (Complexity of functional f_1) Due to the log functions decomposing f_1, the time complexity for computing the functional f_1 directly between 2 decomposable models is

$$\mathcal{O}(n^2 \cdot \omega_{\max} \cdot 2^{\omega_{\max}+1})$$

where $\omega(\mathcal{G})$ is the treewidth of chordal graph $\mathcal{G}$, $\omega_{\max} = \max(\omega(\mathcal{G}_\mathbb{P}), \omega(\mathcal{G}_\mathbb{Q}))$, and n is the number of random variables in $\boldsymbol{X}$.

Proof Recall the $\alpha\beta$-divergence when $\alpha, \beta = 0$:

$$\begin{aligned} D_{AB}^{(0,0)}(\mathbb{P} \parallel \mathbb{Q}) &= f_1(\mathbb{P}, \mathbb{Q}) \\ &= \frac{1}{2} \sum_{x\in\mathcal{X}} (\log \mathbb{P}(\boldsymbol{x}) - \log \mathbb{Q}(\boldsymbol{x}))^2 \\ &= \frac{1}{2} \sum_{x\in\mathcal{X}} \left(\log(\mathbb{P}(\boldsymbol{x}))^2 - 2\left(\log \mathbb{P}(\boldsymbol{x}) \log \mathbb{Q}(\boldsymbol{x})\right) + \log(\mathbb{Q}(\boldsymbol{x}))^2\right) \end{aligned}$$

We will first show how the sum over $\mathcal{X}$ for the term $-2\log(\mathbb{P}(\boldsymbol{x}))\log(\mathbb{Q}(\boldsymbol{x}))$ can be done while avoiding complexity exponential to the number of variables n (i.e. complexity linear to the state space size over all variables $|\mathcal{X}|$). By the end of this process, we will then be able to observe that the complexity for computing

the sum over $\mathcal{X}$ for $\log(\mathbb{P}(\boldsymbol{x}))^2$ and $\log(\mathbb{Q}(\boldsymbol{x}))^2$ is equivalent to or lesser than the complexity for computing the sum over $\mathcal{X}$ for $\log(\mathbb{P}(\boldsymbol{x}))\log(\mathbb{Q}(\boldsymbol{x}))$.

Substituting the MLE for a decomposable model into $\log(\mathbb{P}(\boldsymbol{x}))\log(\mathbb{Q}(\boldsymbol{x}))$, we get:

$$\begin{aligned}
&\sum_{\boldsymbol{x}\in\mathcal{X}} \log \mathbb{P}(\boldsymbol{x}) \log \mathbb{Q}(\boldsymbol{x}) \\
&= \sum_{\boldsymbol{x}\in\mathcal{X}} \left(\sum_{C\in\boldsymbol{C}_{\mathbb{P}}} \log \mathbb{P}_C(\boldsymbol{x}) - \sum_{S\in\boldsymbol{S}_{\mathbb{P}}} \log \mathbb{P}_S(\boldsymbol{x}) \right) \left(\sum_{C'\in\boldsymbol{C}_{\mathbb{Q}}} \log \mathbb{Q}_{C'}(\boldsymbol{x}) - \sum_{S'\in\boldsymbol{S}_{\mathbb{Q}}} \log \mathbb{Q}_{S'}(\boldsymbol{x}) \right) \\
&= \sum_{\boldsymbol{x}\in\mathcal{X}} \left(\sum_{C\in\boldsymbol{C}_{\mathbb{P}}} \sum_{C'\in\boldsymbol{C}_{\mathbb{Q}}} \log \mathbb{P}_C(\boldsymbol{x}) \log \mathbb{Q}_{C'}(\boldsymbol{x}) - \sum_{C\in\boldsymbol{C}_{\mathbb{P}}} \sum_{S'\in\boldsymbol{S}_{\mathbb{Q}}} \log \mathbb{P}_C(\boldsymbol{x}) \log \mathbb{Q}_{S'}(\boldsymbol{x}) - \right. \\
&\qquad \left. \sum_{S\in\boldsymbol{S}_{\mathbb{P}}} \sum_{C'\in\boldsymbol{C}_{\mathbb{Q}}} \log \mathbb{P}_S(\boldsymbol{x}) \log \mathbb{Q}_{C'}(\boldsymbol{x}) + \sum_{S\in\boldsymbol{S}_{\mathbb{P}}} \sum_{S'\in\boldsymbol{S}_{\mathbb{Q}}} \log \mathbb{P}_S(\boldsymbol{x}) \log \mathbb{Q}_{S'}(\boldsymbol{x}) \right)
\end{aligned}$$

All four terms in this sum can be computed by pushing/rearranging the sum over $\mathcal{X}$ to be the inner-most sum and conducting basic variable elimination between 2 terms in the sum:

$\forall(\mathcal{B},\mathcal{D}) \in \{(\boldsymbol{C}_{\mathbb{P}},\boldsymbol{C}_{\mathbb{Q}}),(\boldsymbol{C}_{\mathbb{P}},\boldsymbol{S}_{\mathbb{Q}}),(\boldsymbol{S}_{\mathbb{P}},\boldsymbol{C}_{\mathbb{Q}}),(\boldsymbol{S}_{\mathbb{P}},\boldsymbol{S}_{\mathbb{Q}})\}$:

$$\sum_{\boldsymbol{x}\in\mathcal{X}} \sum_{B\in\mathcal{B}} \sum_{D\in\mathcal{D}} \log \mathbb{P}_B(\boldsymbol{x}) \log \mathbb{Q}_D(\boldsymbol{x}) = |\mathcal{X}_{X-B\cup D}| \sum_{B\in\mathcal{B}} \sum_{D\in\mathcal{D}} \mathrm{VE}(\mathcal{X}, \log \mathbb{P}_B, \log \mathbb{Q}_D)$$

where for $B, D \subseteq \boldsymbol{X}$:

$$\mathrm{VE}(\mathcal{X}, f_D, f_B) = \begin{cases} \left(\sum_{\boldsymbol{x}\in\mathcal{X}_D} f_D(\boldsymbol{x})\right)\left(\sum_{\boldsymbol{x}\in\mathcal{X}_B} f_B(\boldsymbol{x})\right) & D \cap B = \emptyset \\ \sum_{\boldsymbol{x}\in\mathcal{X}_D} f_D(\boldsymbol{x}) f_B(\boldsymbol{x}) & D = B \\ \sum_{\boldsymbol{x}\in\mathcal{X}_D} f_D(\boldsymbol{x}) \sum_{\boldsymbol{x}'\in\mathcal{X}_{B-D}} f_B(\boldsymbol{x}') & B \subset D \\ \sum_{\boldsymbol{x}\in\mathcal{X}_B} f_B(\boldsymbol{x}) \sum_{\boldsymbol{x}'\in\mathcal{X}_{D-B}} f_D(\boldsymbol{x}') & \text{otherwise} \end{cases}$$

The complexity of computing $\mathrm{VE}(\mathcal{X}, f_D, f_B)$ involves determining which case B and D satisfies, which takes time linear to $\mathcal{O}(\max(|B|,|D|))$, and computing the sums, which takes time $\mathcal{O}(2^{\max(|B|,|D|)})$. We also know that $|B|$ and $|D|$ are bounded by the largest clique size in chordal graphs $\mathcal{G}_{\mathbb{P}}$, $\omega(\mathcal{G}_{\mathbb{P}})+1$, and $\mathcal{G}_{\mathbb{Q}}$, $\omega(\mathcal{G}_{\mathbb{Q}})+1$ respectively. Therefore, computing $\mathrm{VE}(\mathcal{X}, f_D, f_B)$ takes complexity $\mathcal{O}(\omega_{\max} \cdot 2^{\omega_{\max}+1})$, where $\omega_{\max} = \max(\omega(\mathcal{G}_{\mathbb{P}}), \omega(\mathcal{G}_{\mathbb{Q}}))$.

Furthermore, when computing $\sum_{\boldsymbol{x}\in\mathcal{X}} \log \mathbb{P}(\boldsymbol{x}) \log \mathbb{Q}(\boldsymbol{x})$, $\mathrm{VE}(\mathcal{X}, f_D, f_B)$ is computed for $(|\boldsymbol{C}_{\mathbb{P}}|+|\boldsymbol{S}_{\mathbb{P}}|)(|\boldsymbol{C}_{\mathbb{Q}}|+|\boldsymbol{S}_{\mathbb{Q}}|)$ distinct Bs and Ds, resulting in a complexity

of:

$$\sum_{x\in\mathcal{X}} \log(\mathbb{P}(x))\log(\mathbb{Q}(x)) \in \mathcal{O}\Big((|\boldsymbol{C}_{\mathbb{P}}| + |\boldsymbol{S}_{\mathbb{P}}|)(|\boldsymbol{C}_{\mathbb{Q}}| + |\boldsymbol{S}_{\mathbb{Q}}|)\cdot \omega_{\max}\cdot 2^{\omega_{\max}+1}\Big)$$

Since the number of cliques and separators in chordal graph $\mathcal{G}$ is bounded by the number of vertices in $\mathcal{G}$, they are also bounded by the number of random variables associated with the graph $\boldsymbol{X}$. In other words:

$$\mathcal{O}(\boldsymbol{C}) \in \mathcal{O}(n)$$
$$\mathcal{O}(\boldsymbol{S}) \in \mathcal{O}(n)$$

therefore:

$$\begin{aligned}\mathcal{O}\Big((|\boldsymbol{C}_{\mathbb{P}}| + |\boldsymbol{S}_{\mathbb{P}}|)(|\boldsymbol{C}_{\mathbb{Q}}| + |\boldsymbol{S}_{\mathbb{Q}}|)\cdot \omega_{\max}\cdot 2^{\omega_{\max}+1}\Big) &\in \mathcal{O}\Big((2n)^2\cdot \omega_{\max}\cdot 2^{\omega_{\max}+1}\Big)\\ &\in \mathcal{O}\Big(n^2\cdot \omega_{\max}\cdot 2^{\omega_{\max}+1}\Big)\end{aligned}$$

Using the same argumentation, the complexity of computing $\sum_{x\in\mathcal{X}} \log(\mathbb{P}(x))^2$ and $\sum_{x\in\mathcal{X}} \log(\mathbb{Q}(x))^2$ is:

$$\sum_{x\in\mathcal{X}} \log(\mathbb{P}(x))^2 \in \mathcal{O}((|\boldsymbol{C}_{\mathbb{P}}| + |\boldsymbol{S}_{\mathbb{P}}|)^2\omega(\mathcal{G}_{\mathbb{P}})2^{\omega(\mathcal{G}_{\mathbb{P}})+1}) \in \mathcal{O}\Big(n^2\cdot \omega_{\max}\cdot 2^{\omega_{\max}+1}\Big)$$
$$\sum_{x\in\mathcal{X}} \log(\mathbb{Q}(x))^2 \in \mathcal{O}((|\boldsymbol{C}_{\mathbb{Q}}| + |\boldsymbol{S}_{\mathbb{Q}}|)^2\omega(\mathcal{G}_{\mathbb{P}})2^{\omega(\mathcal{G}_{\mathbb{P}})+1}) \in \mathcal{O}\Big(n^2\cdot \omega_{\max}\cdot 2^{\omega_{\max}+1}\Big)$$

Therefore, the complexity of computing $D^{0,0}_{AB}(\mathbb{P}\|\mathbb{Q})$ and $f_1(\mathbb{P},\mathbb{Q})$ is:

$$D^{(0,0)}_{AB}(\mathbb{P}\,\|\,\mathbb{Q}) = f_1(\mathbb{P},\mathbb{Q}) \in \mathcal{O}\Big(n^2\cdot \omega_{\max}\cdot 2^{\omega_{\max}+1}\Big)$$

which implies that the functional f_1 can be computed naively while avoiding complexity exponential to the number of variables n.

Since computing f_1 directly is tractable, the focus of the rest of this chapter will be to show how to compute functionals f_2 and f_3 between 2 decomposable models. In order to simplify further exposition, it will be ideal if functionals f_2 and f_3 can be expressed by the more general functional $\mathcal{F}$.

Proposition 1 (f_2 can be expressed in terms of $\mathcal{F}$) f_2 can be expressed in terms of a parameterisation of functional $\mathcal{F}$.

Proof Recall f_2 from Theorem 7:

$$f_2(\mathbb{P},\mathbb{Q};\alpha,\beta) = \sum_{x\in\mathcal{X}} \mathbb{P}(x)^{\alpha}\mathbb{Q}(x)^{\beta}$$

First, set the parameters of $\mathcal{F}$ to be:

$$h[\mathbb{P}](\boldsymbol{x}) = \mathbb{P}(\boldsymbol{x})^{\alpha} \quad g[\mathbb{Q}](\boldsymbol{x}) = \mathbb{Q}(\boldsymbol{x})^{\beta}$$
$$h^*[\mathbb{P}_B] = f^*[\mathbb{P}_B] \quad g^*[\mathbb{Q}_D] = f^*[\mathbb{Q}_D]$$
$$L(x) = \log_b(x)$$

where the logarithmic base b is any real positive number, $\mathcal{G}(S)$ for $S \subseteq \boldsymbol{X}$ is the induced subgraph of $\mathcal{G}$ over the variables in S, and:

$$f^*\left[\mathbb{D}_S\right] = \exp_b\left\{\frac{1}{2} \cdot \frac{1}{|\boldsymbol{C}(\mathcal{G}_{\mathbb{D}})| - |\boldsymbol{C}(\mathcal{G}_{\mathbb{D}}(S))| + 1}\right\}$$

where $\exp_b\{a\} = b^a$, and $S \subseteq \boldsymbol{X}$ such that $\mathcal{G}_{\mathbb{D}}(S)$ is a chordal graph where $\boldsymbol{C}(\mathcal{G}_{\mathbb{D}}) = \boldsymbol{C}(\mathcal{G}_{\mathbb{D}}(S)) \cup \boldsymbol{C}(\mathcal{G}_{\mathbb{D}}(\boldsymbol{X} - S))$

This specific parameterisation of f^* and g^* ensures that they satisfy the requirement in Equation (3.2) from Definition 13 while also removing the log term from the functional $\mathcal{F}$:

$$\begin{aligned}
f^*\left[\prod_{C \in \boldsymbol{C}(\mathcal{G}_{\mathbb{D}})} \mathbb{D}_C\right] = f^*\left[\mathbb{D}_X\right] &= \exp_b\left\{\frac{1}{2} \cdot \frac{1}{|\boldsymbol{C}(\mathcal{G}_{\mathbb{D}})| - |\boldsymbol{C}(\mathcal{G}_{\mathbb{D}}(\boldsymbol{X}))| + 1}\right\} \\
&= \exp_b\left\{\frac{1}{2} \cdot \frac{1}{|\boldsymbol{C}(\mathcal{G}_{\mathbb{D}})| - |\boldsymbol{C}(\mathcal{G}_{\mathbb{D}})| + 1}\right\} \\
&= \exp_b\left\{\frac{1}{2} \cdot \frac{1}{1}\right\} \\
&= b^{1/2} \\
\prod_{C \in \boldsymbol{C}(\mathcal{G}_{\mathbb{D}})} f^*\left[\mathbb{D}_C\right] &= \prod_{C \in \boldsymbol{C}(\mathcal{G}_{\mathbb{D}})} \exp_b\left\{\frac{1}{2} \cdot \frac{1}{|\boldsymbol{C}(\mathcal{G}_{\mathbb{D}})| - |\boldsymbol{C}(\mathcal{G}_{\mathbb{D}}(C))| + 1}\right\} \\
&= \exp_b\left\{\sum_{C \in \boldsymbol{C}(\mathcal{G}_{\mathbb{D}})} \frac{1}{2} \cdot \frac{1}{|\boldsymbol{C}(\mathcal{G}_{\mathbb{D}})| - 1 + 1}\right\} \\
&= \exp_b\left\{\frac{|\boldsymbol{C}(\mathcal{G}_{\mathbb{D}})|}{2|\boldsymbol{C}(\mathcal{G}_{\mathbb{D}})|}\right\} \\
&= b^{1/2} \\
\therefore f^*\left[\prod_{C \in \boldsymbol{C}(\mathcal{G}_{\mathbb{D}})} \mathbb{D}_C\right] &= \prod_{C \in \boldsymbol{C}(\mathcal{G}_{\mathbb{D}})} f^*\left[\mathbb{D}_C\right] = b^{1/2}
\end{aligned}$$

Using these parameters we get:

$$\begin{aligned}
\mathcal{F}(\mathbb{P}, \mathbb{Q}) &= \sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{P}(\boldsymbol{x})^{\alpha} \mathbb{Q}(\boldsymbol{x})^{\beta} \log_b\left(b^{1/2} b^{1/2}\right) \\
&= \sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{P}(\boldsymbol{x})^{\alpha} \mathbb{Q}(\boldsymbol{x})^{\beta} \log_b\left(b^1\right)
\end{aligned}$$

$$= \sum_{x \in \mathcal{X}} \mathbb{P}(\boldsymbol{x})^{\alpha} \mathbb{Q}(\boldsymbol{x})^{\beta}$$

Proposition 2 (f_3 can be expressed in terms of $\mathcal{F}$) f_3 can be expressed in terms of a parameterisation of functional $\mathcal{F}$.

Proof Recall f_3 from Theorem 7:

$$f_3(\mathbb{P}, \mathbb{Q}; \alpha, \beta, c, d) = \sum_{x \in \mathcal{X}} \mathbb{P}(\boldsymbol{x})^{\alpha} \mathbb{Q}(\boldsymbol{x})^{\beta} \log\left(\mathbb{P}(\boldsymbol{x})^{c} \mathbb{Q}(\boldsymbol{x})^{d}\right)$$

Therefore $\mathcal{F} = f_3$ given the following parameters for $\mathcal{F}$:

$$h[\mathbb{P}](x) = \mathbb{P}(x)^{a} \qquad h^{*}[\mathbb{P}](x) = \mathbb{P}(x)^{c}$$
$$g[\mathbb{Q}](x) = \mathbb{Q}(x)^{b} \qquad g^{*}[\mathbb{Q}](x) = \mathbb{Q}(x)^{d}$$
$$L(x) = \log x$$

Therefore, any method that can tractably compute $\mathcal{F}$, as defined in Definition 13, between 2 decomposable models can also tractably compute the $\alpha\beta$-divergence between these models.

With reasoning for the definition of $\mathcal{F}$ established, we can now substitute the joint distributions represented by DMs $\mathbb{P}_{\mathcal{G}_{\mathbb{P}}}$ and $\mathbb{Q}_{\mathcal{G}_{\mathbb{Q}}}$ into functional $\mathcal{F}$. But before we start, first recall the notation established in Section 2.2.3:

$$\mathbb{P}(\boldsymbol{x}) = \prod_{C \in \boldsymbol{C}} \mathbb{P}\left(\boldsymbol{x}_{C - \mathrm{pa}(C)} | \boldsymbol{x}_{\mathrm{pa}(C)}\right) = \prod_{C \in \boldsymbol{C}} \mathbb{P}^{\mathcal{T}}_{C}(\boldsymbol{x}_{C})$$
$$\text{for } C \subset X \; : \; \mathbb{P}^{\mathcal{T}}_{C}\left(\boldsymbol{x}_{X}\right) = \mathbb{P}^{\mathcal{T}}_{C}(\boldsymbol{x}_{C})$$

and $\text{pa}(C)$ is the parent of the maximal clique C in the junction tree of $\mathbb{P}$'s respective chordal graph. Then continuing with the substitution we get:

$$
\begin{aligned}
&\mathcal{F}(\mathbb{P}, \mathbb{Q}; g, h, g^*, h^*) \\
&= \sum_{x \in \mathcal{X}} \Big(g\left[\mathbb{P}\right](x)\Big)\Big(h\left[\mathbb{Q}\right](x)\Big) L\Big(\Big(g^*\left[\mathbb{P}\right](x)\Big)\Big(h^*\left[\mathbb{Q}\right](x)\Big)\Big) \\
&= \sum_{x \in \mathcal{X}} \Big(g\left[\mathbb{P}\right](x)\Big)\Big(h\left[\mathbb{Q}\right](x)\Big) L\left(g^*\left[\prod_{C \in \boldsymbol{C}_{\mathbb{P}}} \mathbb{P}_C^{\mathcal{T}}\right](x) \cdot h^*\left[\prod_{C \in \boldsymbol{C}_{\mathbb{Q}}} \mathbb{Q}_C^{\mathcal{T}}\right](x)\right) \\
&= \sum_{x \in \mathcal{X}} g\left[\mathbb{P}\right](x) h\left[\mathbb{Q}\right](x) L\left(\left(\prod_{C \in \boldsymbol{C}_{\mathbb{P}}} g^*\left[\mathbb{P}_C^{\mathcal{T}}(x)\right]\right) \cdot \left(\prod_{C \in \boldsymbol{C}_{\mathbb{Q}}} h^*\left[\mathbb{Q}_C^{\mathcal{T}}(x)\right]\right)\right) \\
&= \sum_{x \in \mathcal{X}} g\left[\mathbb{P}\right](x) h\left[\mathbb{Q}\right](x) \left[\left(\sum_{C \in \boldsymbol{C}_{\mathbb{P}}} L\left(g^*\left[\mathbb{P}_C^{\mathcal{T}}(x)\right]\right)\right) + \left(\sum_{C \in \boldsymbol{C}_{\mathbb{Q}}} L\left(h^*\left[\mathbb{Q}_C^{\mathcal{T}}(x)\right]\right)\right)\right] \\
&= \left[\sum_{C \in \boldsymbol{C}_{\mathbb{P}}} \sum_{x \in \mathcal{X}} L\left(g^*\left[\mathbb{P}_C^{\mathcal{T}}\right](x)\right)\left(g\left[\mathbb{P}\right](x)\right)\left(h\left[\mathbb{Q}\right](x)\right)\right] + \\
&\quad \left[\sum_{C \in \boldsymbol{C}_{\mathbb{Q}}} \sum_{x \in \mathcal{X}} L\left(h^*\left[\mathbb{Q}_C^{\mathcal{T}}\right](x)\right)\left(g\left[\mathbb{P}\right](x)\right)\left(h\left[\mathbb{Q}\right](x)\right)\right] \\
&= \sum_{C \in \boldsymbol{C}(\mathcal{G}_{\mathbb{P}})} \sum_{x_C \in \mathcal{X}_C} L\left(g^*\left[\mathbb{P}_C^{\mathcal{T}}\right](x_C)\right) SP_C(x_C) + \sum_{C \in \boldsymbol{C}(\mathcal{G}_{\mathbb{Q}})} \sum_{x_C \in \mathcal{X}_C} L\left(h^*\left[\mathbb{Q}_C^{\mathcal{T}}\right](x_C)\right) SP_C(x_C)
\end{aligned}
\tag{3.4}
$$

where, for ease of notation:

$$
\begin{aligned}
SP_C(x_C) &= \sum_{x \in \mathcal{X}_{X-C}} \Big(g\left[\mathbb{P}\right](x_C, x)\Big)\Big(h\left[\mathbb{Q}\right](x_C, x)\Big) \\
&= \sum_{x \in \mathcal{X}_{X-C}} \left[\prod_{C \in \boldsymbol{C}_{\mathbb{P}}} g\left[\mathbb{P}_C^{\mathcal{T}}\right](x_C, x)\right]\left[\prod_{C \in \boldsymbol{C}_{\mathbb{Q}}} h\left[\mathbb{Q}_C^{\mathcal{T}}\right](x_C, x)\right]
\end{aligned}
\tag{3.5}
$$

which represents the marginalisation of all the variables that are not in the clique C over the all the non-log factors produced by $\mathcal{F}$.

As demonstrated, the equalities in Equation (3.4) and Equation (3.5) hold due to a combination of the property of functions g^* and h^* detailed in Definition 13, a basic property of the logarithmic function L, and the associativity of summations.

> **Remark 1** The lower bound complexity of directly computing Equation (3.4) is $\Omega(2^{|X|})$ where $|X|$ is the number of variables. Therefore directly computing the functional $\mathcal{F}$ between 2 high-dimensional decomposable models is intractable.

Proof The complexity of computing Equation (3.4) directly is linear with respect to the product of the cardinality of each sum in the 2 nested sums of Equation (3.4), $\left(|\boldsymbol{C}(\mathcal{G}_{\mathbb{P}})| + |\boldsymbol{C}(\mathcal{G}_{\mathbb{Q}})|\right)|\mathcal{X}|$. Since the lower bound of the number of cliques in $\mathcal{G}_{\mathbb{P}}$ and $\mathcal{G}_{\mathbb{Q}}$ is 1, the lower bound complexity of computing Equation (3.4) is just linear with respect to $|\mathcal{X}|$. However, $|\mathcal{X}|$ is exponential with respect to the number of variables $|\boldsymbol{X}|$. Therefore, the lower bound complexity of naively computing Equation (3.4) is $\Omega(2^{|X|})$, and as a consequence, intractable.

Therefore, in order to compute $\mathcal{F}(\mathbb{P}, \mathbb{Q})$, and as a result $D_{AB}(\mathbb{P}, \mathbb{Q})$, while avoiding complexity exponential to $|\boldsymbol{X}|$, we require a more sophisticated method.

3.2 Multi-Graph Aggregated Sum-Products (MGASPs)

Motivated by the problem of computing Equation (3.4), in this section we will propose a method to compute the sum-product over factors defined over the maximal cliques of 2 chordal graphs, $\mathcal{G}_1$ and $\mathcal{G}_2$. We call this method the MGASP. This method involves first finding a new chordal graph, $\mathcal{H}$, such that $\mathcal{G}_1$ and $\mathcal{G}_2$ are subgraphs of $\mathcal{H}$. We refer to $\mathcal{H}$ as the computation graph of $\mathcal{G}_1$ and $\mathcal{G}_2$. MGASP then assigns the factors defined over maximal cliques of $\mathcal{G}_1$ and $\mathcal{G}_2$ to maximal cliques of $\mathcal{H}$ and runs the JTA on $\mathcal{H}$ with these factors. We show that the procedure that MGASP follows, will result in sum-products equivalent to summing over the product of factors from graphs $\mathcal{G}_1$ and $\mathcal{G}_2$. We then revisit computing Equation (3.4) and apply MGASP to computing it.

Observe from Equation (3.4) that the main source of the computational complexity of $\mathcal{F}(\mathbb{P}, \mathbb{Q})$ is the sum product $SP_C, \forall C \in \boldsymbol{C}(\mathcal{G}_{\mathbb{P}}) \cup \boldsymbol{C}(\mathcal{G}_{\mathbb{Q}})$:

$$SP_C(\boldsymbol{x}_C) = \sum_{\boldsymbol{x} \in \mathcal{X}_{X-C}} \left[\prod_{C \in \boldsymbol{C}_{\mathbb{P}}} g\left[\mathbb{P}_C^{\mathcal{T}}\right](\boldsymbol{x})\right] \left[\prod_{C \in \boldsymbol{C}_{\mathbb{Q}}} h\left[\mathbb{Q}_C^{\mathcal{T}}\right](\boldsymbol{x})\right] \tag{3.5}$$

which has a form that is slightly similar to the final beliefs obtained at each maximal clique of a chordal graph $\mathcal{G}_1$ after running the JTA with factors $\{\psi_C \mid C \in \boldsymbol{C}(\mathcal{G}_1)\}$:

$$\forall C \in \boldsymbol{C}(\mathcal{G}_1) : \beta_C(\boldsymbol{x}_C) = \sum_{\boldsymbol{x} \in \mathcal{X}_{X-C}} \prod_{C \in \boldsymbol{C}(\mathcal{G}_1)} \psi_C(\boldsymbol{x}, \boldsymbol{x}_C) \tag{2.2}$$

Despite their similarities, these sum-products have a significant difference in that the sum-product in Equation (2.2) is a sum over factors defined over maximal cliques of a single chordal graph $\mathcal{G}_1$. However, the sum-product in Equation (3.5) is a sum

over factors defined on the maximal cliques of *two* potentially different chordal graphs, $\mathcal{G}_{\mathbb{P}}$ and $\mathcal{G}_{\mathbb{Q}}$. Therefore for the rest of this section, we will tackle this general problem.

Problem 1 (Sum over factors of 2 chordal graphs) Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be chordal graphs with factors defined over their maximal cliques:

$$\Phi_1 = \left\{ \phi_{1,C} \;\middle|\; C \in \boldsymbol{C}(\mathcal{G}_1) \right\}$$
$$\Phi_2 = \left\{ \phi_{2,C} \;\middle|\; C \in \boldsymbol{C}(\mathcal{G}_2) \right\}$$

We wish to obtain the following sum product for each maximal clique in $\mathcal{G}_1$ and $\mathcal{G}_2$:

$$\forall C \in \boldsymbol{C}(\mathcal{G}_1, \mathcal{G}_2) \;:\; SP_C(\boldsymbol{x}_C) = \sum_{\boldsymbol{x} \in \mathcal{X}_{X-C}} \left[\prod_{C_1 \in \boldsymbol{C}(\mathcal{G}_1)} \phi_{1,C_1}(\boldsymbol{x}_C, \boldsymbol{x}) \right] \left[\prod_{C_2 \in \boldsymbol{C}(\mathcal{G}_2)} \phi_{2,C_2}(\boldsymbol{x}_C, \boldsymbol{x}) \right]$$

However, it can be quite unwieldy to obtain SP_C for maximum cliques in both chordal graphs $\mathcal{G}_1$ and $\mathcal{G}_2$. Instead, we can re-frame Problem 1 as a problem to obtain SP_C for a set of cliques $\boldsymbol{C}$ where all maximal cliques in $\boldsymbol{C}(\mathcal{G}_1)$ and $\boldsymbol{C}(\mathcal{G}_2)$ are either equal to, or a subset of, a clique in $\boldsymbol{C}$. In other words, we need a mapping from cliques in $\boldsymbol{C}(\mathcal{G}_1)$ and $\boldsymbol{C}(\mathcal{G}_2)$ to this hypothetical set of "larger" cliques $\boldsymbol{C}$.

Definition 14 (strictly larger, clique mapping α) A chordal graph $\mathcal{H}$ is *strictly larger* than chordal graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ if all the maximal cliques in both chordal graphs are either a subset of, or equal to, a maximal clique in $\mathcal{H}$. In other words, $\mathcal{H}$ is *strictly larger* than $\mathcal{G}_1$ and $\mathcal{G}_2$ if and only if there exists a mapping α such that:

$$\alpha \;:\; \{\mathcal{G}_1, \mathcal{G}_2\} \to \boldsymbol{C}(\mathcal{G}) \to \boldsymbol{C}(\mathcal{H})$$
$$\text{s.t.} \quad \forall \mathcal{G} \in \{\mathcal{G}_1, \mathcal{G}_2\}, C \in \boldsymbol{C}(\mathcal{G}) \;:\; C \subseteq \alpha(\mathcal{G}, C)$$

Definition 15 (computation graph) If a chordal graph, $\mathcal{H}$, is *strictly larger* than chordal graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, then $\mathcal{H}$ is a *computation graph* of $\mathcal{G}_1$ and $\mathcal{G}_2$.

One direct way to obtain the computation graph $\mathcal{H}$ is to first take the graph union of $\mathcal{G}_1$ and $\mathcal{G}_2$, and then triangulate $\mathcal{G}_1 \cup \mathcal{G}_2$. As stated in Section 2.1.3, choosing a triangulation that results in a minimal treewidth is a **NP**-hard problem, but a valid triangulation can be found with time and memory complexity linear in the number of vertices (Dechter, 2003; Berry et al. 2004; Heggernes, 2006).

Definition 16 (A, cliques assigned by α to C) Assume we have the clique mappings $\alpha : \{\mathcal{G}_1, \mathcal{G}_2\} \to \boldsymbol{C}(\mathcal{G}) \to \boldsymbol{C}(\mathcal{H})$. Then we define a function A to obtain the maximal cliques in chordal graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ mapped by α to some maximal clique in $\mathcal{H}$:

$$A : \{\mathcal{G}_1, \mathcal{G}_2\} \to \boldsymbol{C}(\mathcal{H}) \to \mathcal{P}(\boldsymbol{C}(\mathcal{G}))$$
$$\text{s.t.} \quad \forall \mathcal{G} \in \{\mathcal{G}_1, \mathcal{G}_2\}, C \in \boldsymbol{C}(\mathcal{H}) : A(\mathcal{G}, C) = \{C' : C' \in \boldsymbol{C}(\mathcal{G}) \wedge \alpha_{\mathcal{G}}(C') = C\}$$

where $\mathcal{P}(S)$ is the powerset of set S.

With the concept of a *computation graph* $\mathcal{H}$ and a clique mapping α from maximal cliques in $\mathcal{G}_1$ and $\mathcal{G}_2$ to maximal cliques in $\mathcal{H}$, we shall revisit Problem 1.

Proposition 3 (SP_C is a marginalisation of $SP_{\alpha(\mathcal{G},C)}$) For all $\mathcal{G} \in \{\mathcal{G}_1, \mathcal{G}_2\}, C \in \boldsymbol{C}(\mathcal{G})$:

$$SP_C = \sum_{\boldsymbol{x} \in \mathcal{X}_{\alpha(\mathcal{G},C)-C}} SP_{\alpha(\mathcal{G},C)}(\boldsymbol{x})$$

Proof for all $\mathcal{G} \in \{\mathcal{G}_1, \mathcal{G}_2\}, C \in \boldsymbol{C}(\mathcal{G})$:

$$\begin{aligned}
\sum_{\boldsymbol{x} \in \mathcal{X}_{\alpha(\mathcal{G},C)-C}} SP_{\alpha(\mathcal{G},C)}(\boldsymbol{x}) &= \sum_{\boldsymbol{x} \in \mathcal{X}_{\alpha(\mathcal{G},C)-C}} \sum_{\boldsymbol{x} \in \mathcal{X}_{X-\alpha(\mathcal{G},C)}} \left[\prod_{C_1 \in \boldsymbol{C}(\mathcal{G}_1)} \phi_{1,C_1}(\boldsymbol{x}, \boldsymbol{x})\right] \left[\prod_{C_2 \in \boldsymbol{C}(\mathcal{G}_2)} \phi_{2,C_2}(\boldsymbol{x}, \boldsymbol{x})\right] \\
&= \sum_{\boldsymbol{x} \in \mathcal{X}_{(X-\alpha(\mathcal{G},C))+(\alpha(\mathcal{G},C)-C)}} \left[\prod_{C_1 \in \boldsymbol{C}(\mathcal{G}_1)} \phi_{1,C_1}(\boldsymbol{x})\right] \left[\prod_{C_2 \in \boldsymbol{C}(\mathcal{G}_2)} \phi_{2,C_2}(\boldsymbol{x})\right] \\
&= \sum_{\boldsymbol{x} \in \mathcal{X}_{X-C}} \left[\prod_{C_1 \in \boldsymbol{C}(\mathcal{G}_1)} \phi_{1,C_1}(\boldsymbol{x})\right] \left[\prod_{C_2 \in \boldsymbol{C}(\mathcal{G}_2)} \phi_{2,C_2}(\boldsymbol{x})\right] \\
&= SP_C
\end{aligned}$$

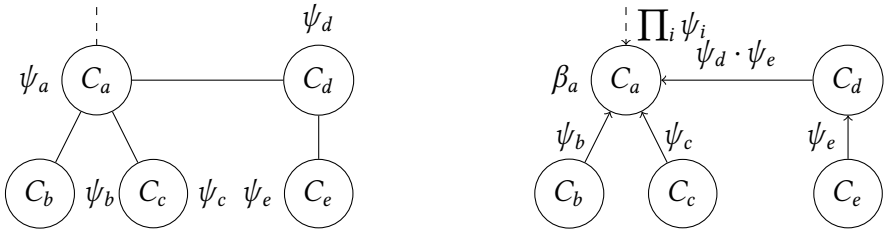
since $\forall \mathcal{G} \in \{\mathcal{G}_1, \mathcal{G}_2\}, C \in \boldsymbol{C}(\mathcal{G}) : C \subseteq \alpha(\mathcal{G}, C)$.

Problem 2 (Sum over factors of 2 chordal graphs (revisited)) Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be chordal graphs with factors defined over their maximal cliques:

$$\Phi_1 = \left\{\phi_{1,C} \,\middle|\, C \in \boldsymbol{C}(\mathcal{G}_1)\right\}$$
$$\Phi_2 = \left\{\phi_{2,C} \,\middle|\, C \in \boldsymbol{C}(\mathcal{G}_2)\right\}$$

and computation graph $\mathcal{H}$ and clique mapping α. We wish to obtain the following sum product for each maximal clique in $\mathcal{G}_1$ and $\mathcal{G}_2$:

$$\forall \mathcal{G} \in \{\mathcal{G}_1, \mathcal{G}_2\}, C \in \boldsymbol{C}(\mathcal{G}) : SP_C(\boldsymbol{x}_C) = \sum_{\boldsymbol{x} \in \mathcal{X}_{\alpha(\mathcal{G},C)-C}} SP_{\alpha(\mathcal{G},C)}(\boldsymbol{x}_C, \boldsymbol{x})$$



$$\psi_i = \prod_{C_1 \in A(\mathcal{G}_1, i)} \phi_{1,C_1} \prod_{C_2 \in A(\mathcal{G}_2, i)} \phi_{2,C_2}$$

$$\beta_a = \psi_a \sum_{\boldsymbol{x} \in \mathcal{X}} \psi_b(\boldsymbol{x}) \psi_c(\boldsymbol{x}) \psi_d(\boldsymbol{x}) \psi_e(\boldsymbol{x}) \prod_i \psi_i(\boldsymbol{x})$$

FIGURE 3.1: Junction tree algorithm on computation graph $\mathcal{H}$ to compute the sum product over the factors Φ_1 and Φ_2 defined over maximal cliques of chordal graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ respectively. We also assume the computation graph $\mathcal{H}$ is not disconnected.

due to Proposition 3. Therefore, what we actually need to obtain is the following sum-product for each maximal clique C^+ in the computation graph $\mathcal{H}$:

$$\forall C' \in \boldsymbol{C}(\mathcal{H}) \; : \; SP_{C'}(\boldsymbol{x}_{C'}) = \sum_{\boldsymbol{x} \in \mathcal{X}_{X - C'}} \left[\prod_{C_1 \in \boldsymbol{C}(\mathcal{G}_1)} \phi_{1,C_1}(\boldsymbol{x}_{C'}, \boldsymbol{x}) \right] \left[\prod_{C_2 \in \boldsymbol{C}(\mathcal{G}_2)} \phi_{2,C_2}(\boldsymbol{x}_{C'}, \boldsymbol{x}) \right] \tag{3.6}$$

and to marginalise these sum-products to obtain $SP_C, \forall C \in \boldsymbol{C}(\mathcal{G}_1, \mathcal{G}_2)$.

For the rest of this section, we will provide details on how we can use the JTA on $\mathcal{H}$ using a set of specifically constructed factors to solve Problem 2. Specifically, in Section 3.2.1 we will first show how to construct this set of specific initial factors and how to use them with the JTA on a computation graph $\mathcal{H}$ that is not disconnected. Then in Section 3.2.2, we will extend this to handling cases when $\mathcal{H}$ is disconnected. Finally, in Section 3.3 we will go back to our original problem of computing $\mathcal{F}(\mathbb{P}, \mathbb{Q})$ and apply the methods developed in this section to it. Discussions about computational complexity will be deferred to Section 3.4.

3.2.1 $\mathcal{H}$ is a Connected Graph

Since $SP_C, \forall C \in \boldsymbol{C}(\mathcal{H})$ are just sum products over the factors

$$\Phi = \left\{ \phi_{1,C} \;\middle|\; C \in \boldsymbol{C}(\mathcal{G}_1) \right\} \bigcup \left\{ \phi_{2,C} \;\middle|\; C \in \boldsymbol{C}(\mathcal{G}_2) \right\}$$

as defined in Problem 2, we can use the JTA over the computation graph of $\mathcal{G}_1$ and $\mathcal{G}_2$, $\mathcal{H}$, with factors Φ to obtain SP_C for each maximal clique in $\boldsymbol{C}(\mathcal{H})$.

Theorem 9 (SP_C can be obtained from JTA) Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be chordal graphs with factors defined over their maximal cliques, Φ_1 and Φ_2, and a computation graph $\mathcal{H}$. Then, let Ψ be the set of factors formed by the product of factors assigned to each maximal clique in $\mathcal{H}$.

$$\Psi := \left\{ \prod_{C_1 \in A(\mathcal{G}_1, C)} \phi_{1,C1} \prod_{C_2 \in A(\mathcal{G}_2, C)} \phi_{2,C2} \; : \; C \in \boldsymbol{C}(\mathcal{H}) \right\}$$

After running the junction tree algorithm over the junction tree of $\mathcal{H}$ with factors Ψ, we will get the following beliefs over each maximal clique in $\mathcal{H}$:

$$\begin{aligned} \forall C \in \boldsymbol{C}(\mathcal{H}) \; : \; \beta_C(\boldsymbol{x}_C) &= SP_C(\boldsymbol{x}_C) \\ &= \sum_{\boldsymbol{x} \in \mathcal{X}_{X-C}} \left[\prod_{C_1 \in \boldsymbol{C}(\mathcal{G}_1)} \phi_{1,C_1}(\boldsymbol{x}) \right] \left[\prod_{C_2 \in \boldsymbol{C}(\mathcal{G}_2)} \phi_{2,C_2}(\boldsymbol{x}) \right] \end{aligned}$$

Proof Recall from Section 2.2.4 that given a chordal graph $\mathcal{H}$ and a set of factors Φ, the JTA provides the following belief for each maximal clique in the junction tree/forest, $\mathcal{T} = (\boldsymbol{C}(\mathcal{H}), \boldsymbol{S}(\mathcal{H}))$, of the chordal graph $\mathcal{H}$:

$$\forall C \in \boldsymbol{C} \; : \; \beta_C(\boldsymbol{x}_C) = \sum_{\boldsymbol{x} \in \mathcal{X}_{X-C}} \prod_{\phi \in \Phi} \phi(\boldsymbol{x}, \boldsymbol{x}_C)$$

Therefore, by using the set of factors Ψ for the JTA:

$$\Psi := \left\{ \prod_{C_1 \in A(\mathcal{G}_1, C)} \phi_{1,C1} \prod_{C_2 \in A(\mathcal{G}_2, C)} \phi_{2,C2} \; : \; C \in \boldsymbol{C}(\mathcal{H}) \right\}$$

we directly obtain the following beliefs by simple substitution:

$$\begin{aligned} \forall C \in \boldsymbol{C} \; : \; \beta_C(\boldsymbol{x}_C) &= \sum_{\boldsymbol{x} \in \mathcal{X}_{X-C}} \prod_{\psi \in \Psi} \psi(\boldsymbol{x}, \boldsymbol{x}_C) \\ &= \sum_{\boldsymbol{x} \in \mathcal{X}_{X-C}} \prod_{C \in \boldsymbol{C}(\mathcal{H})} \prod_{C_1 \in A(\mathcal{G}_1, C)} \phi_{1,C1}(\boldsymbol{x}, \boldsymbol{x}_C) \prod_{C_2 \in A(\mathcal{G}_2, C)} \phi_{2,C2}(\boldsymbol{x}, \boldsymbol{x}_C) \\ &= \sum_{\boldsymbol{x} \in \mathcal{X}_{X-C}} \prod_{C_1 \in \boldsymbol{C}(\mathcal{G}_1)} \phi_{1,C1}(\boldsymbol{x}, \boldsymbol{x}_C) \prod_{C_2 \in \boldsymbol{C}(\mathcal{G}_2)} \phi_{2,C2}(\boldsymbol{x}, \boldsymbol{x}_C) \\ &= SP_C(\boldsymbol{x}_C) \end{aligned}$$

since $\forall \mathcal{G} \in \{\mathcal{G}_1, \mathcal{G}_2\} \; : \; \bigcup_{C \in \boldsymbol{C}(\mathcal{H})} A(\mathcal{G}, C) = \boldsymbol{C}(\mathcal{G})$.

3.2.2 $\mathcal{H}$ is a Disconnected Graph

In Section 3.2.1, we assumed that the computation graph for chordal graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, $\mathcal{H}$ is not disconnected. However, this might not always be the case. In this section, we will show how the methods in Section 3.2.1 can be extended to handle cases where $\mathcal{H}$ is disconnected.

When the computation graph $\mathcal{H}$ is disconnected, $\mathcal{H}$ can be represented as a list of chordal graphs, $\mathcal{H} = \{\mathcal{H}_i\}$. Therefore, we also have a list of junction trees for each chordal graph in $\mathcal{H}$, $\mathcal{T} = \{\mathcal{T}_i\}$, as well.

Definition 17 (τ, clique to junction tree mapping) Let τ be a mapping from the maximal cliques of $\mathcal{H}$ to a junction tree in $\mathcal{T}$ that contains the maximal clique:

$$\tau : \boldsymbol{C}(\mathcal{H}) \to \mathcal{T}$$
$$\text{s.t.} \quad \forall C \in \boldsymbol{C}(\mathcal{H}) : C \in \boldsymbol{C}(\tau(C))$$

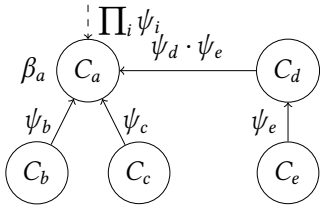
Since the disconnected computation graph, $\mathcal{H}$, is still strictly larger than the chordal graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, by Definition 14, there is still a mapping α from maximal cliques in $\mathcal{G}_1$ and $\mathcal{G}_2$ to maximal cliques in $\mathcal{H}$. However, since chordal graph $\mathcal{H}$ is now comprised of multiple chordal graphs, and therefore junction trees, we are unable to apply Theorem 9 directly to compute *SP* from Equation (3.5). The reason for this is because there is no single junction tree to run the junction tree algorithm on, therefore factors from different junction trees are unable to propagate to each other.

Instead, we show in Theorem 10, that having a disconnected computation graph $\mathcal{H}$, and therefore a set of junction trees, $\mathcal{T}$, which are disconnected from each other, essentially decomposes *SP* into smaller sub-problems over each junction tree in $\mathcal{T}$. The results of these sub-problems can then be combined via multiplication to compute *SP*. An illustration of the result from Theorem 10 and its difference in computing *SP* on a fully connected $\mathcal{H}$ can be found in Figure 3.2.

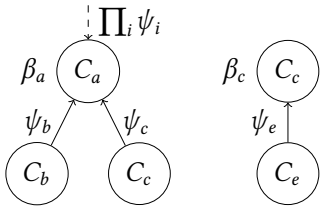
Theorem 10 (SP_C for disconnected $\mathcal{H}$) If the computation graph $\mathcal{H}$ for chordal graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ is disconnected, $SP_C, \forall C \in \boldsymbol{C}(\mathcal{H})$ in Equation (3.6) can be re-expressed as follows:

$$SP_C(\boldsymbol{x}_C) = \beta_C(\boldsymbol{x}_C) \prod_{\mathcal{T}_i \in \mathcal{T} - \tau(C)} \sum_{\boldsymbol{x} \in \mathcal{X}_{c(\mathcal{T}_i)}} \beta_{c(\mathcal{T}_i)}(\boldsymbol{x}_C, \boldsymbol{x})$$

where $c(\mathcal{T}_i)$ represents any clique in the set of maximal cliques in junction tree $\mathcal{T}_i$.



(a) Junction Tree Computation



(b) Junction Forest Computation

Figure 3.2: Differences in getting the clique beliefs over each maximal clique in the computation graph $\mathcal{H}$ between a fully connected and a disconnected computation graph.

Proof Recall from Equation (3.6), $\forall C \in \boldsymbol{C}(\mathcal{H})$:

$$SP_C(\boldsymbol{x}_C) = \sum_{\boldsymbol{x} \in \mathcal{X}_{X-C}} \left[\prod_{C_1 \in \boldsymbol{C}(\mathcal{G}_1)} \phi_{1,C_1}(\boldsymbol{x}_C, \boldsymbol{x}) \right] \left[\prod_{C_2 \in \boldsymbol{C}(\mathcal{G}_2)} \phi_{2,C_2}(\boldsymbol{x}_C, \boldsymbol{x}) \right] \tag{3.6}$$

Following the usual steps to initiate the junction tree algorithm, we define the set of initial clique beliefs, Ψ, to use in the junction tree algorithm as such:

$$\Psi := \left\{ \prod_{C_1 \in A(\mathcal{G}_1, C)} \phi_{1,C1} \prod_{C_2 \in A(\mathcal{G}_2, C)} \phi_{2,C2} \;\middle|\; C \in \boldsymbol{C}(\mathcal{H}) \right\}$$

Then $SP_C, C \in \boldsymbol{C}(\mathcal{H})$ can be re-expressed in terms of these initial clique beliefs ψ:

$$SP_C(\boldsymbol{x}_C) = \sum_{\boldsymbol{x} \in \mathcal{X}_{X-C}} \prod_{C' \in \boldsymbol{C}(\mathcal{H})} \psi_{C'}(\boldsymbol{x}_C, \boldsymbol{x})$$

Due to the independence between maximal cliques of different junction trees in the junction forest $\boldsymbol{\mathcal{T}}$, we can split the sum in $SP_C, C \in \boldsymbol{C}(\mathcal{H})$ into a product of sum-products over each junction tree in $\boldsymbol{\mathcal{T}}$:

$$\begin{aligned}
&SP_C(\boldsymbol{x}_C) \\
&= \sum_{\boldsymbol{x} \in \mathcal{X}_{X-C}} \prod_{C' \in \boldsymbol{C}(\mathcal{H})} \psi_{C'}(\boldsymbol{x}_C, \boldsymbol{x}) \\
&= \sum_{\boldsymbol{x} \in \mathcal{X}_{X-C}} \prod_{T \in \boldsymbol{\mathcal{T}}} \prod_{C' \in \boldsymbol{C}(T)} \psi_{C'}(\boldsymbol{x}_C, \boldsymbol{x}) \\
&= \sum_{\boldsymbol{x}_{T_1} \in \mathcal{X}_{\boldsymbol{C}(T_1)}} \cdots \sum_{\boldsymbol{x}_{\tau(C)} \in \mathcal{X}_{\boldsymbol{C}(\tau(C))-C}} \cdots \sum_{\boldsymbol{x}_{T_{|\boldsymbol{\mathcal{T}}|}} \in \mathcal{X}_{\boldsymbol{C}\left(T_{|\boldsymbol{\mathcal{T}}|}\right)}} \prod_{T \in \boldsymbol{\mathcal{T}}} \prod_{C' \in \boldsymbol{C}(T)} \psi_{C'}(\boldsymbol{x}_{T_1}, \ldots, \boldsymbol{x}_{\tau(c)}, \ldots, \boldsymbol{x}_{T_{|\boldsymbol{\mathcal{T}}|}}) \\
&= \left(\sum_{\boldsymbol{x}_{\tau(C)} \in \mathcal{X}_{\boldsymbol{C}(\tau(C))-C}} \prod_{C' \in \boldsymbol{C}(\tau(C))} \psi_{C'}(\boldsymbol{x}_{\tau(C)}) \right) \prod_{T \in \boldsymbol{\mathcal{T}} - \tau(C)} \left(\sum_{\boldsymbol{x}_T \in \mathcal{X}_{\boldsymbol{C}(T)}} \prod_{C' \in \boldsymbol{C}(T)} \psi_{C'}(\boldsymbol{x}_T) \right) \\
&= \beta_C(\boldsymbol{x}_C) \prod_{T \in \boldsymbol{\mathcal{T}} - \tau(C)} \sum_{\boldsymbol{x} \in \mathcal{X}_{C(T)}} \beta_{C(T)}(\boldsymbol{x}_C, \boldsymbol{x})
\end{aligned}$$

where $\boldsymbol{\mathcal{T}}_i$ is the i-th junction tree in the set of junction trees $\boldsymbol{\mathcal{T}}$.

Therefore, all we need to do in order to obtain the required beliefs is to run the junction tree algorithm for each junction tree in $\boldsymbol{\mathcal{T}}$ separately.

Once the computation for each junction tree in $\boldsymbol{\mathcal{T}}$ is finished, we can compute $SP_C(\boldsymbol{x}_C)$ for all $C \in \boldsymbol{C}(\mathcal{H})$. Therefore, even if the computation graph $\mathcal{H}$ is disconnected, we can compute SP_C for all $C \in \boldsymbol{C}(\mathcal{H})$.

3.3 Using MGASP to Compute $\mathcal{F}(\mathbb{P}, \mathbb{Q})$

Now that we have established MGASP to compute the sum-product over factors of multiple chordal graphs, we shall return to our original problem of computing $\mathcal{F}(\mathbb{P}, \mathbb{Q})$ from Equation (3.4). Specifically, in this section, we will show how to construct a specific set of factors over maximal cliques of $\mathcal{G}_{\mathbb{P}}$ and $\mathcal{G}_{\mathbb{Q}}$ from $\mathcal{F}(\mathbb{P}, \mathbb{Q})$ such that using MGASP with this set of factors results in efficiently obtaining the sum-products SP_C for each maximal clique C in the computation graph of $\mathcal{G}_{\mathbb{P}}$ and $\mathcal{G}_{\mathbb{Q}}$. We then show how these sum-products are used in the computation of $\mathcal{F}(\mathbb{P}, \mathbb{Q})$.

Recall we want to compute $\mathcal{F}(\mathbb{P}, \mathbb{Q})$ as expressed in Equation (3.4):

$$\begin{aligned}\mathcal{F}(\mathbb{P}, \mathbb{Q}) = &\sum_{C \in \boldsymbol{C}(\mathcal{G}_{\mathbb{P}})} \sum_{\boldsymbol{x}_C \in \mathcal{X}_C} L\left(g^*\left[\mathbb{P}_C^{\mathcal{T}}\right](\boldsymbol{x}_C)\right) SP_C(\boldsymbol{x}_C) + \\ &\sum_{C \in \boldsymbol{C}(\mathcal{G}_{\mathbb{Q}})} \sum_{\boldsymbol{x}_C \in \mathcal{X}_C} L\left(h^*\left[\mathbb{Q}_C^{\mathcal{T}}\right](\boldsymbol{x}_C)\right) SP_C(\boldsymbol{x}_C)\end{aligned} \tag{3.4}$$

where

$$SP_C(\boldsymbol{x}_C) = \sum_{\boldsymbol{x} \in \mathcal{X}_{X-C}} \left[\prod_{C \in \boldsymbol{C}_{\mathbb{P}}} g\left[\mathbb{P}_C^{\mathcal{T}}\right](\boldsymbol{x}_C, \boldsymbol{x})\right]\left[\prod_{C \in \boldsymbol{C}_{\mathbb{Q}}} h\left[\mathbb{Q}_C^{\mathcal{T}}\right](\boldsymbol{x}_C, \boldsymbol{x})\right] \tag{3.5}$$

From Section 3.2 we know that $\mathcal{G} \in \{\mathcal{G}_{\mathbb{P}}, \mathcal{G}_{\mathbb{Q}}\}, C \in \boldsymbol{C}(\mathcal{G}) : SP_{\alpha(\mathcal{G},C)}$ can be computed using the JTA over a computation graph $\mathcal{H}$ for $\mathcal{G}_{\mathbb{P}}$ and $\mathcal{G}_{\mathbb{Q}}$ using the following factors:

$$\Psi := \left\{ \prod_{C \in A(\mathcal{G}_{\mathbb{P}}, C)} g\left[\mathbb{P}_C^{\mathcal{T}}\right] \prod_{C \in A(\mathcal{G}_{\mathbb{Q}}, C)} h\left[\mathbb{Q}_C^{\mathcal{T}}\right] \;\middle|\; C \in \boldsymbol{C}(\mathcal{H}) \right\} \tag{3.7}$$

However, observe that the 2 nested sums in $\mathcal{F}(\mathbb{P}, \mathbb{Q})$ have innermost sums over $\mathcal{X}_C$ with similar forms but over different sets of maximal cliques, $\boldsymbol{C}(\mathcal{G}_{\mathbb{P}})$ and $\boldsymbol{C}(\mathcal{G}_{\mathbb{Q}})$ respectively. Therefore, using Theorem 11, we can re-express $\mathcal{F}(\mathbb{P}, \mathbb{Q})$ such that the sum over $\mathcal{X}_C$ of both nested sums are over the same set of maximal cliques, $\boldsymbol{C}(\mathcal{H})$ which will allow us to use the method outlined in Section 3.2.

Theorem 11 (Mapping SP_C to maximal cliques of $\mathcal{H}$) Assume we have 2 decomposable models $\mathbb{P}_{\mathcal{G}_{\mathbb{P}}}$ and $\mathbb{Q}_{\mathcal{G}_{\mathbb{Q}}}$ and a computation graph $\mathcal{H}$ for both models. By Definition 14, we also have a mapping α from maximal cliques in $\mathbb{P}_{\mathcal{G}_{\mathbb{P}}}$ and $\mathbb{Q}_{\mathcal{G}_{\mathbb{Q}}}$ to maximal cliques in $\mathcal{H}$. Then the following equivalence holds for

$(\mathbb{D}, f) \in \{(\mathbb{P}, g^*), (\mathbb{Q}, h^*)\}$:

$$\sum_{C \in \boldsymbol{C}(\mathcal{G}_\mathbb{D})} \sum_{\boldsymbol{x}_C \in \mathcal{X}_C} L\left(f^*\left[\mathbb{D}_C^{\mathcal{T}}\right](\boldsymbol{x}_C)\right) SP_C(\boldsymbol{x}_C) = \sum_{C \in \boldsymbol{C}(\mathcal{G}_\mathbb{D})} \sum_{\substack{\boldsymbol{x}_{\alpha(\mathcal{G}_\mathbb{D},C)} \in \\ \mathcal{X}_{\alpha(\mathcal{G}_\mathbb{D},C)}}} L\left(f^*\left[\mathbb{D}_C^{\mathcal{T}}\right](\boldsymbol{x}_{\alpha(\mathcal{G}_\mathbb{D},C)})\right) SP_{\alpha(\mathcal{G}_\mathbb{D},C)}(\boldsymbol{x}_{\alpha(\mathcal{G}_\mathbb{D},C)}) \tag{3.8}$$

Proof Starting from the left hand side of Equation (3.8), we can split the innermost sum into 2 sums, one over the set $\mathcal{X}_{\alpha(\mathcal{G}_\mathbb{D},C)-C}$ and $\mathcal{X}_{X-\alpha(\mathcal{G}_\mathbb{D},C)}$, then move the sum over $\mathcal{X}_{\alpha(\mathcal{G}_\mathbb{D},C)-C}$ outside and merge it with the sum over $\mathcal{X}_C$:

$$\begin{aligned}
&\sum_{C \in \boldsymbol{C}_\mathbb{D}} \sum_{\boldsymbol{x}_C \in \mathcal{X}_C} L\left(g^*\left[\mathbb{D}_C^{\mathcal{T}}\right](\boldsymbol{x}_C)\right) SP_C(\boldsymbol{x}_C) \\
&= \sum_{C \in \boldsymbol{C}_\mathbb{D}} \sum_{\boldsymbol{x}_C \in \mathcal{X}_C} L\left(g^*\left[\mathbb{D}_C^{\mathcal{T}}\right](\boldsymbol{x}_C)\right) \sum_{\boldsymbol{x} \in \mathcal{X}_{X-C}} \left(g\left[\mathbb{D}\right](\boldsymbol{x}_C, \boldsymbol{x})\right)\left(h\left[\mathbb{Q}\right](\boldsymbol{x}_C, \boldsymbol{x})\right) \\
&= \sum_{C \in \boldsymbol{C}_\mathbb{D}} \sum_{\boldsymbol{x}_C \in \mathcal{X}_C} L\left(g^*\left[\mathbb{D}_C^{\mathcal{T}}\right](\boldsymbol{x}_C)\right) \sum_{\boldsymbol{x} \in \mathcal{X}_{X-\alpha(\mathcal{G}_\mathbb{D},C)+\alpha(\mathcal{G}_\mathbb{D},C)-C}} \left(g\left[\mathbb{D}\right](\boldsymbol{x}_C, \boldsymbol{x})\right)\left(h\left[\mathbb{Q}\right](\boldsymbol{x}_C, \boldsymbol{x})\right) \\
&= \sum_{C \in \boldsymbol{C}_\mathbb{D}} \sum_{\boldsymbol{x}_C \in \mathcal{X}_C} \sum_{\boldsymbol{x}' \in \mathcal{X}_{\alpha(\mathcal{G}_\mathbb{D},C)-C}} L\left(g^*\left[\mathbb{D}_C^{\mathcal{T}}\right](\boldsymbol{x}_C, \boldsymbol{x}')\right) \\
&\qquad \sum_{\boldsymbol{x} \in \mathcal{X}_{X-\alpha(\mathcal{G}_\mathbb{D},C)}} \left(g\left[\mathbb{D}\right](\boldsymbol{x}_C, \boldsymbol{x}, \boldsymbol{x}')\right)\left(h\left[\mathbb{Q}\right](\boldsymbol{x}_C, \boldsymbol{x}, \boldsymbol{x}')\right) \\
&= \sum_{C \in \boldsymbol{C}_\mathbb{D}} \sum_{\boldsymbol{x}_{\alpha(\mathcal{G}_\mathbb{D},C)} \in \mathcal{X}_{\alpha(\mathcal{G}_\mathbb{D},C)}} L\left(g^*\left[\mathbb{D}_C^{\mathcal{T}}\right](\boldsymbol{x}_{\alpha(\mathcal{G}_\mathbb{D},C)})\right) \\
&\qquad \sum_{\boldsymbol{x} \in \mathcal{X}_{X-\alpha(\mathcal{G}_\mathbb{D},C)}} \left(g\left[\mathbb{D}\right](\boldsymbol{x}_{\alpha(\mathcal{G}_\mathbb{D},C)}, \boldsymbol{x})\right)\left(h\left[\mathbb{Q}\right](\boldsymbol{x}_{\alpha(\mathcal{G}_\mathbb{D},C)}, \boldsymbol{x})\right) \\
&= \sum_{C \in \boldsymbol{C}_\mathbb{D}} \sum_{\boldsymbol{x}_{\alpha(\mathcal{G}_\mathbb{D},C)} \in \mathcal{X}_{\alpha(\mathcal{G}_\mathbb{D},C)}} L\left(g^*\left[\mathbb{D}_C^{\mathcal{T}}\right](\boldsymbol{x}_{\alpha(\mathcal{G}_\mathbb{D},C)})\right) SP_{\alpha(\mathcal{G}_\mathbb{D},C)}(\boldsymbol{x}_{\alpha(\mathcal{G}_\mathbb{D},C)})
\end{aligned}$$

Thanks to Theorem 11, we can re-express the functional $\mathcal{F}$ between 2 decomposable models in Equation 3.4 as such:

$$\mathcal{F}(\mathbb{P}_{\mathcal{G}_\mathbb{P}}, \mathbb{Q}_{\mathcal{G}_\mathbb{Q}}) = \left[\sum_{C \in \boldsymbol{C}(\mathcal{G}_\mathbb{P})} \sum_{\substack{\boldsymbol{x}_{\alpha(\mathcal{G}_\mathbb{P},C)} \in \\ \mathcal{X}_{\alpha(\mathcal{G}_\mathbb{P},C)}}} L\left(g^*\left[\mathbb{P}_C^{\mathcal{T}}(\boldsymbol{x}_{\alpha(\mathcal{G}_\mathbb{P},C)})\right]\right) SP_{\alpha(\mathcal{G}_\mathbb{P},C)}(\boldsymbol{x}_{\alpha(\mathcal{G}_\mathbb{P},C)})\right] + \left[\sum_{C \in \boldsymbol{C}(\mathcal{G}_\mathbb{Q})} \sum_{\substack{\boldsymbol{x}_{\alpha(\mathcal{G}_\mathbb{Q},C)} \in \\ \mathcal{X}_{\alpha(\mathcal{G}_\mathbb{Q},C)}}} L\left(h^*\left[\mathbb{Q}_C^{\mathcal{T}}(\boldsymbol{x}_{\alpha(\mathcal{G}_\mathbb{Q},C)})\right]\right) SP_{\alpha(\mathcal{G}_\mathbb{Q},C)}(\boldsymbol{x}_{\alpha(\mathcal{G}_\mathbb{Q},C)})\right]$$

With that, we will now obtain all instances of $SP_{\alpha(\mathcal{G},C)}, \mathcal{G} \in \{\mathcal{G}_\mathbb{P}, \mathcal{G}_\mathbb{Q}\}$ by running the JTA over the junction tree of $\mathcal{H}$ with the set of initial factors Ψ in Equation (3.7).

Therefore, using the junction tree algorithm, we obtain beliefs over each maximal clique in the chordal graph $\mathcal{H}$ that we can use to substitute for SP in Equation (3.4):

$$\begin{aligned}\mathcal{F}(\mathbb{P},\mathbb{Q}) = &\sum_{C\in\boldsymbol{C}(\mathcal{G}_\mathbb{P})}\sum_{\substack{\boldsymbol{x}_{\alpha(\mathcal{G}_\mathbb{P},C)}\in\\ \mathcal{X}_{\alpha(\mathcal{G}_\mathbb{P},C)}}} L\left(g^*\left[\mathbb{P}_C^{\mathcal{T}}\right](\boldsymbol{x}_{\alpha(\mathcal{G}_\mathbb{P},C)})\right)\beta_{\alpha(\mathcal{G}_\mathbb{P},C)}(\boldsymbol{x}_{\alpha(\mathcal{G}_\mathbb{P},C)})+\\ &\sum_{C\in\boldsymbol{C}(\mathcal{G}_\mathbb{Q})}\sum_{\substack{\boldsymbol{x}_{\alpha(\mathcal{G}_\mathbb{Q},C)}\in\\ \mathcal{X}_{\alpha(\mathcal{G}_\mathbb{Q},C)}}} L\left(h^*\left[\mathbb{Q}_C^{\mathcal{T}}\right](\boldsymbol{x}_{\alpha(\mathcal{G}_\mathbb{Q},C)})\right)\beta_{\alpha(\mathcal{G}_\mathbb{Q},C)}(\boldsymbol{x}_{\alpha(\mathcal{G}_\mathbb{Q},C)})\end{aligned} \tag{3.9}$$

Now that the main computationally heavy part of computing $\mathcal{F}(\mathbb{P},\mathbb{Q})$ is done, we can compute the sums in Equation (3.9) directly. Section 3.4 will show that the complexity of computing the expression in Equation (3.9), and in general, the complexity of using MGASP to compute $\mathcal{F}$ between 2 decomposable models, is $\mathcal{O}\left(|\boldsymbol{X}|\cdot 2^{\omega(\mathcal{H})}\right)$, where $\mathcal{H}$ is the computation graph of $\mathcal{G}_\mathbb{P}$ and $\mathcal{G}_\mathbb{Q}$. Therefore, using MGASP is more efficient than $\mathcal{O}(2^{X-C})$, the complexity of computing $\mathcal{F}$ directly, when $\mathcal{H}$ is not a fully saturated graph.

3.4 Complexity of Computing $\alpha\beta$-Divergence between Joint Distributions of Decomposable Models

We can determine the computational complexity of computing the $\alpha\beta$-divergence between 2 decomposable models by first checking what the given values for α and β are. This step takes $\mathcal{O}(1)$ time. When $\alpha,\beta=0$, from Theorem 8 we know that the complexity of computing $D_{AB}^{0,0}(\mathbb{P}\|\mathbb{Q})$ is:

$$D_{AB}^{0,0}(\mathbb{P},\mathbb{Q})\in\mathcal{O}(|\boldsymbol{X}|^2\cdot\omega_{\max}\cdot 2^{\omega_{\max}+1})$$

where $\omega(\mathcal{G})$ is the treewidth of some chordal graph $\mathcal{G}$ and $\omega_{\max} = \max(\omega(\mathcal{G}_\mathbb{P}),\omega(\mathcal{G}_\mathbb{Q}))$.

When α and β take values other than 0, we require the use of MGASP, as described in Section 3.2, to compute parameterisations of $\mathcal{F}$ between the decomposable models. In general, the $\alpha\beta$-divergence is a linear combination of different parameterisations of $\mathcal{F}$. Therefore, the complexity of computing the $\alpha\beta$-divergence is equivalent to computing $\mathcal{F}$ in big-O notation. As such, for the remainder of this section, we will discuss the overall complexity of MGASP for computing $\mathcal{F}$ between 2 decomposable models.

The first step of MGASP involves assigning factors constructed from the conditional probability tables over the maximal cliques in models $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$ and $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$ to maximal cliques in the computation graph $\mathcal{H}$. Therefore, for each factor ψ, and therefore for

each maximal clique in $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$ and $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$, we will need to search through each maximal clique in $\mathcal{H}$ in the worst case, to find a suitable clique to assign the factor ψ to. This results in the following complexity:

$$\mathcal{O}(|\boldsymbol{C}(\mathcal{G}_\mathbb{P})| \cdot |\boldsymbol{C}(\mathcal{H})|) + \mathcal{O}(|\boldsymbol{C}(\mathcal{G}_\mathbb{P})| \cdot |\boldsymbol{C}(\mathcal{H})|) \in \mathcal{O}(|\boldsymbol{X}|^2)$$

since the number of maximal cliques in any chordal graph is bounded by the number of vertices in the graph.

Once all the relevant factors have been assigned to their respective maximal cliques in $\mathcal{H}$, we need to run the junction tree algorithm to calibrate the clique tree/forest of $\mathcal{H}$ with these factors. The complexity of the junction tree algorithm is:

$$\mathcal{O}(|\boldsymbol{C}(\mathcal{H})| \cdot 2^{\omega(\mathcal{H})+1}) \in \mathcal{O}(|\boldsymbol{X}| \cdot 2^{\omega(\mathcal{H})+1})$$

Once the clique tree/forest is calibrated and we know $\beta_{\alpha(C)}$ for all $C \in \boldsymbol{C}(\mathcal{G}_\mathbb{P}) \cup \boldsymbol{C}(\mathcal{G}_\mathbb{Q})$, we can compute Equation (3.9):

$$\begin{aligned}
\mathcal{F}(\mathbb{P}, \mathbb{Q}) = &\sum_{C \in \boldsymbol{C}(\mathcal{G}_\mathbb{P})} \sum_{\substack{\boldsymbol{x}_{\alpha(C)} \in \\ \mathcal{X}_{\alpha(C)}}} L\left(g^*\left[\mathbb{P}_C^{\mathcal{T}}\right](\boldsymbol{x}_{\alpha(C)})\right) \beta_{\alpha(C)}(\boldsymbol{x}_{\alpha(C)}) + \\
&\sum_{C \in \boldsymbol{C}(\mathcal{G}_\mathbb{Q})} \sum_{\substack{\boldsymbol{x}_{\alpha(C)} \in \\ \mathcal{X}_{\alpha(C)}}} L\left(h^*\left[\mathbb{Q}_C^{\mathcal{T}}\right](\boldsymbol{x}_{\alpha(C)})\right) \beta_{\alpha(C)}(\boldsymbol{x}_{\alpha(C)}) \\
&\in \mathcal{O}(\boldsymbol{C}(\mathcal{G}_\mathbb{P}) \cdot 2^{\omega(\mathcal{H})+1}) + \mathcal{O}(\boldsymbol{C}(\mathcal{G}_\mathbb{Q}) \cdot 2^{\omega(\mathcal{H})+1}) \\
&\in \mathcal{O}(|\boldsymbol{X}| \cdot 2^{\omega(\mathcal{H})+1})
\end{aligned}$$

Adding up the computational complexity of each step in MGASP results in the final complexity of computing the functional $\mathcal{F}$ between 2 decomposable models $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$ and $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$:

$$\mathcal{O}(|\boldsymbol{X}|^2) + \mathcal{O}(|\boldsymbol{X}| \cdot 2^{\omega(\mathcal{H})+1}) + \mathcal{O}(|\boldsymbol{X}| \cdot 2^{\omega(\mathcal{H})+1}) \in \mathcal{O}(|\boldsymbol{X}| \cdot 2^{\omega(\mathcal{H})+1})$$

Therefore, the computational complexity of computing the $\alpha\beta$-divergence between $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$ and $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$ is:

$$D_{AB}^{(\alpha,\beta)}(\mathbb{P} \| \mathbb{Q}) \in \begin{cases} \mathcal{O}(|\boldsymbol{X}|^2 \cdot \omega_{\max} \cdot 2^{\omega_{\max}+1}) & \alpha, \beta = 0 \\ \mathcal{O}(|\boldsymbol{X}| \cdot 2^{\omega(\mathcal{H})+1}) & \text{otherwise} \end{cases} \tag{3.10}$$

3.5 Runtime Comparison with Existing Method

Recall that a method already exists for computing the KL divergence between 2 BNs (Moral et al. 2021) which we will refer to as `mcgo`. Also note that it is possible to take a distribution represented by a BN and, in exchange for some loss

	time (seconds)		
Network	mcgo	MGASP	Time Reduction
cancer	0.0153	**0.0108**	29.41 %
earthquake	**0.0112**	**0.0112**	0.00 %
survey	0.0157	**0.0103**	34.39 %
asia	0.0193	**0.0179**	7.25 %
sachs	0.0500	**0.0170**	66.00 %
child	0.0848	**0.0444**	47.64 %
insurance	0.4400	**0.1694**	61.50 %
water	12.0979	**8.1683**	32.48 %
mildew	**22.5723**	24.9018	−9.35 %
alarm	0.3586	**0.0974**	72.84 %
hailfinder	0.9455	**0.1868**	80.24 %
hepar2	1.4656	**0.2556**	82.56 %
win95pts	1.1361	**0.4073**	64.15 %
barley	-	**6.4252**	-

TABLE 3.1: Runtimes for the mcgo and MGASP on computing the KL divergence between 2 BNs. Faster times are in bold.

in independence information, represent it using a DM instead (Koller & Friedman, 2009, p.p. 134). Therefore, one might ask: how does the practical runtime of MGASP compare to mcgo when computing the KL divergence between 2 BNs?

Unfortunately, Moral et al. did not provide a theoretical characterisation of the runtime complexity of their approach. Therefore, to answer this question, we will instead replicate the experiment used by Moral et al. They chose a set of BNs from the *bnlearn* (Scutari, 2010) repository (https://www.bnlearn.com/bnrepository/) to sample from and estimated a second BN from these samples. The authors have provided these *estimated* BNs for each of the BNs from *bnlearn* used in their experiments in their code repository: https://github.com/mgomez-olmedo/KL-pgmpy. Therefore, we will use this set of BN from their repository in our own experiments.

Now that we have multiple pairs of BNs, one original and one estimated from samples, we then compute the KL divergence between each BN pair using both mcgo and MGASP. We repeat this 10 times in order to get an estimate of both methods' runtime in seconds. We also do not factor in the conversion of these BNs into DMs in the final runtime.

Theoretically, both methods should compute the exact same value for the KL divergence between each network pair. However, practically, discrepancies can occur due to how each method handles cases where the KL divergence is theoretically undefined (i.e. cases where the probability of a cell is non-zero in $\mathbb{P}$ but zero in $\mathbb{Q}$).

We run the experiments on an Intel NUC-10i7FNH with 64GB of RAM. The implementation of both methods are in Python and use the pgmpy library (Ankan & Panda, 2015). The repository for the implementation for our proposed method is https://gitlab.com/lklee/div-comp. The results on the "barley" network for mcgo are missing due to a lack of available memory on the system.

From the results in Table 3.1, we can observe that despite mcgo containing numerous computation optimisations, our direct application of belief propagation to carry out the computation has a practical runtime that is comparable to mcgo. Furthermore, on some networks, MGASP is faster than mcgo, probably due to having a lower overhead and being better able to leverage the optimized code in the pgmpy library for the bulk of the computation.

3.6 Case Study: KL Divergence in Model Selection

Although allowing for a simpler implementation that can leverage existing library implementations of the junction tree algorithm for most of the computation is a satisfactory result by itself, recall that the original motivation of MGASP is to compute a wider range of divergences between graphical models. Therefore, in order to motivate the need of using divergences other than the KL divergence, in this section we present a case study on the application of computing divergences between BNs for the problem of model selection.

Consider a scientist who, in an attempt to model a natural phenomenon that they have samples from, constructs 2 candidate BNs, A and B, either from theory or some other expertise about the phenomenon. They then wish to determine, using the samples, which candidate model is a better representation of the phenomenon they wish to model. One way to do this, is to estimate a new BN, E, from the samples and compute the divergence between E and the candidate models.

In order to recreate this scenario synthetically, we use the BN sachs from the bnlearn repository (Scutari, 2010) as the "phenomenon" the scientist wishes to model. The scientist's "candidate models" are then constructed by removing edges from sachs and marginalising the CPTs according to (Choi et al. 2005).

Specifically candidate model A is obtained by removing the following directed edges:

- PKA → Raf
- PKC → PKA
- Plcg → PIP3

while candidate model B is obtained by removing the following directed edges:

- PKC → Raf
- PKC → Mek
- PKA → Mek

On the deletion of an edge $Y \to X$, the CPT of X is marginalised as follows (Choi et al. 2005) where $\boldsymbol{Z} = \text{parents}(X) \setminus Y$:

$$\mathbb{P}'_{X|\boldsymbol{Z}}(x|\boldsymbol{z}) := \sum_{y \in \text{Dom}(Y)} \mathbb{P}_{X|Y,\boldsymbol{Z}}(x|y,\boldsymbol{z})\mathbb{P}_Y(y)$$

Sampling 100000 samples from sachs, we then learn BN E from these samples using the constraint-based structure learner in *pgmpy* and *maximum likelihood estimation* with Laplace smoothing for learning the parameters of E. The use of a smoothing technique is to ensure that the KL divergence is defined. We then compute the Hellinger and KL divergence between the candidate models and the estimated model: $D(\text{A}||\text{E})$ and $D(\text{B}||\text{E})$. We repeat the experiment 20 times, with different random samples from sachs.

From the results in Table 3.2, we can observe that the KL divergence indicates that A is the BN closest to E and that the scientist should choose A, while the Hellinger distance indicates the opposite, choosing B instead. With this discrepancy, the question then is, which candidate model, A or B, is actually the closer approximation to the actual phenomenon, and therefore, which divergence is correct.

Since, for the purpose of this case study, we already have the true model of the "phenomenon" we are modelling, we can just compute the divergence between our candidate models and sachs to get an answer.

From Table 3.3, we can observe that when computing the divergence from the candidate models and sachs, both divergences agree that B is closer to sachs. Consequently, in our case study, our scientist would have chosen the incorrect model if they only used the KL divergence in their experiment.

Of course, it might be possible to avoid such a scenario if a different smoothing technique is used to learn the parameters of E. However, the use of multiple divergences is still needed in order for the scientist to even be aware of possible issues in the smoothing technique used in the first place. In general, the main takeaway from this example should be that, one must not be over-reliant on just a single divergence, and the use of a wide array of divergences can be helpful in avoiding mistakes in model selection.

run	Kullback-Leibler		Hellinger	
	$A\|\|E$	$B\|\|E$	A, E	B, E
0	**0.4034**	0.5420	0.3030	**0.2927**
1	**0.3989**	0.5116	0.3008	**0.2894**
2	**0.3976**	0.5167	0.3014	**0.2905**
3	**0.4015**	0.5153	0.3021	**0.2904**
4	**0.3993**	0.5215	0.3017	**0.2908**
5	**0.4017**	0.5260	0.3027	**0.2918**
6	**0.3962**	0.5153	0.3007	**0.2897**
7	**0.4060**	0.5088	0.3036	**0.2888**
8	**0.3971**	0.5252	0.3008	**0.2914**
9	**0.4021**	0.5243	0.3027	**0.2904**
10	**0.3973**	0.5411	0.3013	**0.2928**
11	**0.4050**	0.5142	0.3036	**0.2890**
12	**0.4016**	0.5042	0.3023	**0.2889**
13	**0.4022**	0.5140	0.3030	**0.2888**
14	**0.4013**	0.5227	0.3024	**0.2911**
15	**0.4000**	0.5172	0.3021	**0.2891**
16	**0.4045**	0.5111	0.3032	**0.2896**
17	**0.3967**	0.5120	0.3014	**0.2893**
18	**0.4041**	0.5126	0.3029	**0.2898**
19	**0.4080**	0.5140	0.3045	**0.2907**
mean	0.4012	0.5185	0.3023	0.2902
std. err.	0.0033	0.0097	0.0010	0.0012

TABLE 3.2: Divergence between the candidate models and a Bayesian network estimated from 20 randomly sampled datasets of size 10000. Lower time in bold.

	$A\|\|$sachs	$B\|\|$sachs
Kullback-Leibler	0.3687	**0.3090**
Hellinger	0.3013	**0.2921**

TABLE 3.3: Divergence between the candidate models and the original Bayesian network sachs. Lower time in bold.

3.7 Conclusion

In conclusion, we showed how computing the functional $\mathcal{F}$, and therefore the $\alpha\beta$-divergence, between the joint distribution represented by 2 DMs is equivalent to belief propagation on a clique tree/forest with a set of specific initial factors defined based on how the joint distribution of DMs decomposes the functional $\mathcal{F}$. The result is a method for computing the sum-product over factors from multiple chordal graphs, multi-graph aggregated sum-product (MGASP), with complexity exponential to the treewidth of the computation graph $\mathcal{H}$ of the given chordal graphs. Therefore, using MGASP is more efficient than computing the $\mathcal{F}$ between the DMs directly unless $\mathcal{H}$ is a fully saturated graph.

Another advantage of MGASP over previous methods is that it can be easily implemented in any environment that has a preexisting implementation of the JTA. Furthermore, since MGASP is capable of computing the general functional $\mathcal{F}$ between the joint distribution of 2 DMs, it is capable of computing, or even providing an approximation, of other divergences or functionals, and not just the $\alpha\beta$-divergence.

Chapter 4

Computing Divergence between Marginal Distributions

High-dimensionality can negatively impact the ability to analyse and understand occurrences of concept drift/shift between multiple high-dimensional distributions. In fact, if we were to take 2 distributions and measure the divergence between them, but for only a subset of variables within these distributions, we will find that as we increase the number of variables within this subset, the divergence measured between the distributions will only increase or stay the same (Webb et al. 2018). Therefore, as dimensionality increases, we are increasing likely to encounter situations where the marginal distribution over only a small subspace, i.e. variable subset, consistently changes, causing the divergence between the joint distributions to be consistently high. Therefore, the ability to measure the divergence between the subspaces of two high-dimensional distributions is vital for the analysis and understanding of distributional changes in high dimensions.

However, the problem with trying to compute the divergence between the marginal distributions of two DMs is that the marginal distributions don't necessarily have an immediate decomposition similar to the joint distribution of a DM.

To better illustrate this problem, first recall that the joint distribution of a DM $\mathbb{D}_{\mathcal{G}_\mathbb{D}}$ decomposes based on its maximal cliques and minimal separators:

$$\mathbb{D}_X = \frac{\prod_{C \in \boldsymbol{C}(\mathcal{G}_\mathbb{D})} \mathbb{D}_C}{\prod_{S \in \boldsymbol{S}(\mathcal{G}_\mathbb{D})} \mathbb{D}_S} = \prod_{C \in \boldsymbol{C}(\mathcal{G}_\mathbb{D})} \mathbb{D}_C^{\mathcal{I}}$$

Then the marginal distribution over variables $\boldsymbol{Z} \subset \boldsymbol{X}$ represented by DM $\mathbb{D}_{\mathcal{G}_\mathbb{D}}$ is obtained by marginalising the variables $\boldsymbol{Y} = \boldsymbol{X} \setminus \boldsymbol{Z}$:

$$\mathbb{D}_Z = \sum_{\boldsymbol{y} \in \mathcal{Y}} \mathbb{D}_X(\boldsymbol{y}) = \sum_{\boldsymbol{y} \in \mathcal{Y}} \prod_{C \in \boldsymbol{C}(\mathcal{G}_\mathbb{D})} \mathbb{D}_C^{\mathcal{I}}(\boldsymbol{y}) \tag{4.1}$$

resulting in the marginal distribution $\mathbb{D}_Z$ with the domain $\mathcal{Z}$ which grows exponentially with respect to the number of variables in $\boldsymbol{Z}$. Therefore, in situations

where $|Z|$ is large, which is likely in high-dimensional problems, working with $\mathbb{D}_Z$ directly is infeasible.

In order to avoid this problem, we need to find a decomposition of the marginal distribution $\mathbb{D}_Z$ that will allow for a more compact representation and which decomposes $\mathcal{F}$ into smaller sub-problems. Such a decomposition will involve splitting $\mathbb{D}_Z$ into a product of smaller independent marginalising sums. More formally, we want to find a decomposition, for marginal distributions $\mathbb{D}_Z \in \{\mathbb{P}_Z, \mathbb{Q}_Z\}$, of the following form:

$$\mathbb{D}_Z = \sum_{Y \in \mathcal{Y}} \prod_{C \in \boldsymbol{C}(\mathcal{G})} \mathbb{D}_C^{\mathcal{T}}(\boldsymbol{y}) = \prod_{B \in \mathcal{B}} \sum_{\boldsymbol{y}_B \in \mathcal{Y}_B} \prod_{C \in \boldsymbol{C}(\mathcal{G}(B))} \mathbb{D}_B^{\mathcal{T}}(\boldsymbol{y}_B) \tag{4.2}$$

for some set $\mathcal{B}$ where $\forall B \in \mathcal{B} : B \subseteq V(\mathcal{G}_{\mathbb{D}})$. Therefore, we shall tackle the problem of finding a decomposition for a marginal distribution $\mathbb{D}_Z$ of the DM $\mathbb{D}_{\mathcal{G}_{\mathbb{D}}}$ in Section 4.1.

However, in order to use the methods described in Chapter 3 to assist in computing the marginal divergence between DMs $\mathbb{P}_{\mathcal{G}_{\mathbb{P}}}$ and $\mathbb{Q}_{\mathcal{G}_{\mathbb{Q}}}$, we require the decomposition of $\mathbb{P}_Z$ and $\mathbb{Q}_Z$ to be expressed in terms of a product of factors defined over the maximal clique of some chordal graph, or in other words, for a DM $\mathbb{D}_{\mathcal{G}} \in \{\mathbb{P}_{\mathcal{G}_{\mathbb{P}}}, \mathbb{Q}_{\mathcal{G}_{\mathbb{Q}}}\}$ we want to express $\mathbb{D}_Z$ in terms of

$$\mathbb{D}_Z = \sum_{Y \in \mathcal{Y}} \prod_{C \in \boldsymbol{C}(\mathcal{G})} \mathbb{D}_C^{\mathcal{T}}(\boldsymbol{y}) = \prod_{C' \in \boldsymbol{C}(\mathcal{G}')} \phi_{C'} \tag{4.3}$$

We will show how to find the chordal graph $\mathcal{G}'$ and factors $\left\{\phi_C : C \in \boldsymbol{C}(\mathcal{G}')\right\}$ where such a decomposition is possible in Section 4.2.

Then in Section 4.3, we show how to use the constructs developed throughout this chapter to compute the $\alpha\beta$-divergence between the marginal distributions of two DMs.

We wrap up the technical portion of this chapter in Section 4.4 with a discussion on the computational complexity of the method developed in this chapter and how its computational complexity varies greatly depending on the set of marginal variables Z. We then conclude and summarise this chapter in Section 4.5

4.1 Decomposing Marginal Distributions of a Decomposable Model

In order to tackle the problem of finding a decomposition of marginal distributions encoded in a DM with the form in Equation (4.2), we will first introduce concepts and tools that will help us further break down and better conceptualise the problem. The first of these concepts is a partition of the chordal graph's maximal cliques

that also partitions the variables we wish to sum out, Y. This partition, which we will call the $\mathcal{N}$-partition, will help us split the sum in Equation (4.1) into a product of smaller sums over the variables in each partition. This then achieves the decomposition set out in Equation (4.2).

Definition 18 ($\mathcal{N}$-partitions) Let $\mathcal{G}$ be a chordal graph with clique tree/forest $\mathcal{T}(\mathcal{G})$, and $Y \subset X$ be a subset of the variables in $\mathcal{G}$. Then any set, $\mathcal{N}_{\mathcal{G},Y}$, that is an $\mathcal{N}$-partition of $\mathcal{G}$ and Y, has the following properties:

1. The elements of $\mathcal{N}_{\mathcal{G},Y}$ are subsets of the vertices of $\mathcal{G}$,
$$\forall N \in \mathcal{N}_{\mathcal{G},Y} \,:\, N \subset V(\mathcal{G})$$
2. $\{\boldsymbol{C}(\mathcal{G}(N)) \mid N \in \mathcal{N}_{\mathcal{G},Y}\}$ is a partition of the maximal cliques of $\mathcal{G}$,
$$\bigcup_{N \in \mathcal{N}_{\mathcal{G},Y}} \boldsymbol{C}(\mathcal{G}(N)) = \boldsymbol{C}(\mathcal{G})$$
$$\forall N, N' \in \mathcal{N}_{\mathcal{G},Y} \,:\, N \neq N' \Rightarrow \boldsymbol{C}(\mathcal{G}(N)) \cap \boldsymbol{C}(\mathcal{G}(N')) = \emptyset$$
3. $\{Y \cap X_N \mid N \in \mathcal{N}_{\mathcal{G},Y}, Y \cap X_N \neq \emptyset\}$ is a partition of Y,
$$\bigcup_{N \in \mathcal{N}_{\mathcal{G},Y}} Y \cap X_N = Y$$
$$\forall N, N' \in \mathcal{N}_{\mathcal{G},Y} \,:\, N \neq N' \Rightarrow Y \cap X_N \cap X_{N'} = \emptyset$$
4. $\{\mathcal{T}(\mathcal{G}(N)) \mid N \in \mathcal{N}_{\mathcal{G},Y}\}$ are subtrees of the junction tree/forest $\mathcal{T}(\mathcal{G})$

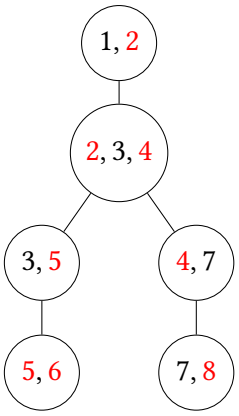
Therefore, for any chordal graph $\mathcal{G}$ and variable subset Y, the trivial $\mathcal{N}_{\mathcal{G},Y} = \{V(\mathcal{G})\}$ will always exist. Note that there might be sets in $\mathcal{N}_{\mathcal{G},Y}$ that do not contain any vertices for Y. Furthermore, although $\mathcal{N}_{\mathcal{G},Y}$ partitions the variables in Y, it does not partition all the variables in X.

The way we will obtain these partitions in this thesis is to take the junction tree/forest of $\mathcal{G}$ and remove any minimal separators that do not contain any variables in Y. This procedure results in multiple trees, each representing a single partition N_i.

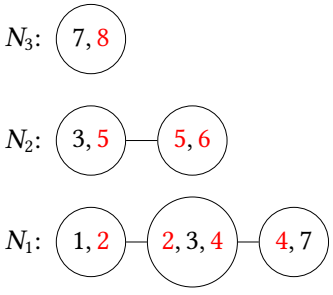
Example 1 Consider a chordal graph $\mathcal{G}$ with the junction tree in Figure 4.1a with $Y = \{2, 4, 5, 6, 8\}$. In order to construct a $\mathcal{N}$-partition for $\mathcal{G}$ with variables Y, $\mathcal{N}_{\mathcal{G},Y}$, we can remove the edges in the junction tree of $\mathcal{G}$ that do not contain vertices associated with variables in Y. Specifically, in the junction tree in Figure 4.1a, we can only remove 2 edges:

1. the edge between the cliques $(2, 3, 4)$ and $(3, 5)$ as the edge between them only contains vertex 3 and $X_3 \notin Y$,
2. the edge between the cliques $(4, 7)$ and $(7, 8)$ as the edge between them only contains vertex 7 and $X_7 \notin Y$.

Removing these two edges from the junction tree of $\mathcal{G}$ results in three separately connected subtrees, and therefore three sets in the partition $\mathcal{N}_{\mathcal{G},Y}$ as we can see in Figure 4.1b.



(a) Clique tree of a chordal graph.



(b) One possible $\mathcal{N}$-partition for variables of interest.

Figure 4.1: A clique tree and possible $\mathcal{N}$-partitions for the variables highlighted in red.

With the concept of $\mathcal{N}$-partitions established, we can now revisit the problem in Equation (4.2) of finding a decomposition of over probability distributions modelled by decomposable models $\mathbb{D}_Z \in \{\mathbb{P}_Z, \mathbb{Q}_Z\}$. First, for the sake of brevity, let:

$$\begin{aligned} \mathcal{N}_{\mathbb{D},Y} &:= \mathcal{N}_{\mathcal{G}_\mathbb{D},Y} \\ \forall N_i \in \mathcal{N}_{\mathbb{D},Y} &: \boldsymbol{W}^{(i)} = \boldsymbol{Y} \cap \boldsymbol{X}_{N_i} \end{aligned}$$

then

$$\begin{aligned} \mathbb{D}_Z &= \sum_{\boldsymbol{y} \in \mathcal{Y}} \prod_{C \in \boldsymbol{C}(\mathcal{G}_\mathbb{D})} \mathbb{D}_C^{\mathcal{T}}(\boldsymbol{y}) \\ &= \sum_{\boldsymbol{y} \in \mathcal{Y}} \prod_{N \in \mathcal{N}_{\mathbb{D},Y}} \prod_{C \in \boldsymbol{C}(\mathcal{G}_\mathbb{D}(N))} \mathbb{D}_C^{\mathcal{T}}(\boldsymbol{y}) \\ &= \prod_{N \in \mathcal{N}_{\mathbb{D},Y}} \sum_{\boldsymbol{w}^{(i)} \in \mathcal{W}^{(i)}} \prod_{C \in \boldsymbol{C}(\mathcal{G}_\mathbb{D}(N))} \mathbb{D}_C^{\mathcal{T}}(\boldsymbol{w}^{(i)}) \end{aligned}$$

due to $\mathcal{N}_{\mathcal{G}_\mathbb{D},Y}$ partitioning $\boldsymbol{Y}$ and $\boldsymbol{C}(\mathcal{G}_\mathbb{D})$ as defined in Definition 18. For ease of notation, let us define the function $\boldsymbol{\varphi}$ to act as a shorthand for the marginalising sum over $\mathcal{W}^{(i)}$.

Definition 19 (Marginal $\mathcal{N}$-probability, φ) Let $\mathbb{D}_{\mathcal{G}_\mathbb{D}}$ be a decomposable model, $N \subset V(\mathcal{G}_\mathbb{D})$ be a subset of vertices in $\mathcal{G}_\mathbb{D}$, and $\boldsymbol{Y} \subset \boldsymbol{X}$ be a subset of variables in $\mathbb{D}_{\mathcal{G}_\mathbb{D}}$. Then we define a functional, φ, that returns a factor that is equivalent to summing over the variables $\boldsymbol{Y}$ out of the product over the CPTs of $\boldsymbol{C}(\mathcal{G}_\mathbb{D}(N))$.

$$\varphi_{N-Y}[\mathbb{D}] = \sum_{\boldsymbol{w} \in \mathcal{W}} \prod_{C \in \boldsymbol{C}(\mathcal{G}_\mathbb{D}(N))} \mathbb{D}_C^{\mathcal{T}}(\boldsymbol{w})$$

where

$$\boldsymbol{W} = \boldsymbol{Y} \cap \boldsymbol{X}_N$$

and through a slight abuse of notation

$$N - \boldsymbol{Y} := N \setminus V(\boldsymbol{Y}) = N \setminus V(\boldsymbol{W})$$

as in $N - \boldsymbol{Y}$ is the set of vertices in N minus the vertices associated with the variables $\boldsymbol{Y}$.

Note that it is possible for the marginal $\mathcal{N}$-probability $\varphi_{N-\boldsymbol{Y}}$ to be defined over the empty set when $N - \boldsymbol{Y} = \{\} \Leftrightarrow \boldsymbol{W} = \boldsymbol{X}_N$. In such situations, $\varphi_{N-\boldsymbol{Y}}$ is just a scalar factor, i.e. a number in $\mathbb{R}$.

With this in mind, we shall define a function $\boldsymbol{\varphi}$ that returns a set of marginal $\mathcal{N}$-probabilities given an $\mathcal{N}$-partition $\mathcal{N}_{\mathcal{G},\boldsymbol{Y}}$:

$$\boldsymbol{\varphi}(\mathcal{N}_{\mathcal{G},\boldsymbol{Y}}) := \left\{ \varphi_{N-\boldsymbol{Y}} \;\middle|\; N \in \mathcal{N}_{\mathcal{G},\boldsymbol{Y}} \right\} \tag{4.4}$$

In order to fully illustrate how the marginal $\mathcal{N}$-probability φ can be computed, let us consider an example of finding the marginal $\mathcal{N}$-probability of a set in the $\mathcal{N}$-partition in Figure 4.1b, specifically set N_2 in Figure 4.1b, $\varphi_{N_2-\boldsymbol{Y}}$.

Example 2 Consider the $\mathcal{N}$-partition in Figure 4.1b for the chordal graph described in Example 1. The marginal $\mathcal{N}$-probability $\varphi_{N_2-\boldsymbol{Y}}$ is:

$$\varphi_{N_2-\boldsymbol{Y}}[\mathbb{D}](\boldsymbol{x}_3) = \sum_{\boldsymbol{w} \in \mathcal{X}_{5,6}} \mathbb{D}^{\mathcal{T}}_{3,5}(\boldsymbol{w}, \boldsymbol{x}_3)\mathbb{D}^{\mathcal{T}}_{5,6}(\boldsymbol{w}, \boldsymbol{x}_3)$$

Therefore, obtaining the factor $\varphi_{N_2-\boldsymbol{Y}}$ requires carrying out the sum over $\mathcal{X}_{5,6}$ for each values in $\mathcal{X}_3$, thus having a complexity of $\mathcal{O}(2^3)$. That said, by borrowing ideas from belief propagation on junction trees, the sum in $\varphi_{N_2-\boldsymbol{Y}}$ can then be decomposed

$$\varphi_{N_2-\boldsymbol{Y}}[\mathbb{D}](\boldsymbol{x}_5) = \sum_{\boldsymbol{w} \in \mathcal{X}_5} \mathbb{D}^{\mathcal{T}}_{3,5}(\boldsymbol{w}, \boldsymbol{x}_5) \sum_{\boldsymbol{w} \in \mathcal{X}_6} \mathbb{D}^{\mathcal{T}}_{5,6}(\boldsymbol{w}, \boldsymbol{x}_5)$$

resulting in both an inner and outer sum that has a computational complexity of $\mathcal{O}(2^2)$ for similar reasons. Therefore, this decomposition results in a lower complexity in obtaining $\varphi_{N_2-\boldsymbol{Y}}$.

With the decomposition in Example 2 in mind, one might ask: what is the worst case computational complexity for obtaining the marginal $\mathcal{N}$-probability for some set in a $\mathcal{N}$-partition and some marginal variable set $\boldsymbol{Z} = \boldsymbol{X} - \boldsymbol{Y}$ of a DM?

Proposition 4 (Worst case complexity of obtaining φ) Let N be some vertex set from an $\mathcal{N}$-partition and $\boldsymbol{Y} \subset \boldsymbol{X}$ be the variables we wish to

marginalise out of the joint distribution of a DM. Then the worst case computational complexity for obtaining the marginal $\mathcal{N}$-probability φ_{N-Y} is

$$\mathcal{O}(2^{|N|})$$

Proof In general we know that the upper bound for the complexity of obtaining some marginal $\mathcal{N}$-probability φ_{N-Y} is $\mathcal{O}(2^{|N|})$ since computing it directly will require us to:

1. iterate through each possible value in $\mathcal{X}_{N-Y}$, requiring a complexity of $\mathcal{O}(2^{|N-Y|})$
2. for each value $z \in \mathcal{X}_{N-Y}$, sum over all values in $\mathcal{X}_Y$, requiring a complexity of $\mathcal{O}(2^{|Y|})$

But as we can see from Example 2, decomposition of the marginalising sum can reduce the computational complexity by orders of magnitude. However, we will now show that there can be situations where such a decomposition is not possible, resulting in a the same complexity as the upper bound we provided earlier, $\mathcal{O}(2^{|N|})$.

Consider a decomposable model with the chordal graph structure $\mathcal{G} = (\{1,2,3\}, \{(1,2),(2,3)\})$. Assume that in this example $Y = \{2\}$ and that we want to find the marginal $\mathcal{N}$-probability $\varphi_{\{1,2,3\}-\{2\}}$.

$$\varphi_{\{1,2,3\}-\{2\}}[\mathbb{D}](\boldsymbol{x}_{1,3}) = \sum_{\boldsymbol{w} \in \mathcal{X}_2} \mathbb{D}^{\mathcal{T}}_{1,2}(\boldsymbol{w}, \boldsymbol{x}_{1,3}) \mathbb{D}^{\mathcal{T}}_{2,3}(\boldsymbol{w}, \boldsymbol{x}_{1,3})$$

As we can observe, no decomposition of the sum in $\varphi_{\{1,2,3\}-\{2\}}$ can occur since the sum is only over the domain of a single variable, $\boldsymbol{X}_2$. Therefore, the final complexity of obtaining $\varphi_{\{1,2,3\}-\{2\}}$

$$\mathcal{O}(2^3) = \mathcal{O}(2^{|\{1,2,3\}|})$$

the same complexity as the upper bound complexity we provided at the start of this proof. Therefore, the worst case complexity of computing the marginal sum for any set N in an $\mathcal{N}$-partition is

$$\mathcal{O}(2^{|N|})$$

Now that we have established the concept of a marginal $\mathcal{N}$-probability φ, we can use the set of marginal $\mathcal{N}$-probabilities of an $\mathcal{N}$-partition, $\boldsymbol{\varphi}$, to express $\mathbb{D}_Z$ in a

more compact manner:

$$\mathbb{D}_Z = \prod_{N \in \boldsymbol{\mathcal{N}}_{\mathbb{D},Y}} \varphi_{N-Y}[\mathbb{D}] \tag{4.5}$$

4.2 Reframing the Problem

Although we now have a decomposition of the marginal distribution $\mathbb{D}_Z$ encoded in a DM $\mathbb{D}_\mathcal{G}$, the ability to compute the $\alpha\beta$-divergence between these decomposed marginal distributions is still not clear. Recall that the methods to compute the joint distribution of two DMs in Chapter 3 require chordal graphs with factors defined over the maximal cliques of each chordal graph. Therefore, in order to use the methods in Chapter 3 to compute $\mathbb{D}_Z$, we need to find an expression of the decomposition of $\mathbb{D}_Z$ that is equivalent to a product of factors over some arbitrary chordal graph $\mathcal{G}'$. Specifically, in this section, we will show how to obtain an expression of $\mathbb{D}_Z$ where this arbitrary chordal graph $\mathcal{G}'$ has maximal cliques based on the sets in an $\boldsymbol{\mathcal{N}}$-partition of $\mathcal{G}$ for marginal variables $\boldsymbol{Z} \subset \boldsymbol{X}$, and factors that are the set of marginal $\boldsymbol{\mathcal{N}}$-probabilities $\boldsymbol{\varphi}$ from Definition 19.

The first step we can take to try and obtain the needed alternate chordal graph $\mathcal{G}'$ is to formalise the idea of constructing a new chordal graph where the graph's maximal cliques are the sets in an $\boldsymbol{\mathcal{N}}$-partition. Furthermore, since SP_{N-Y} and the factors it sums over are not defined over any variables in $\boldsymbol{Y}$, when constructing this new graph, we should also remove any vertices associated with variables in $\boldsymbol{Y}$. Therefore, we will define a new function Γ that will help give us the graphs we need given an $\boldsymbol{\mathcal{N}}$-partition.

Definition 20 ($\boldsymbol{\mathcal{N}}$-graphs, $\Gamma(A, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y})$) Let $\mathcal{G}$ be a chordal graph with an $\boldsymbol{\mathcal{N}}$-partition over variables $\boldsymbol{Y}$, $\boldsymbol{\mathcal{N}}_{\mathcal{G},Y}$, and $A \subseteq \boldsymbol{X}$ be some subset of all the variables $\boldsymbol{X}$. Then we shall define the function Γ that returns a graph where its maximal cliques are partitions in $\boldsymbol{\mathcal{N}}_{\mathcal{G},Y}$ but with only the vertices associated with the variables in A:

$$V(\Gamma(A, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y})) = V(A) \bigcap \left(\bigcup \boldsymbol{\mathcal{N}}_{\mathcal{G},Y} \right)$$
$$E(\Gamma(A, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y})) = \left\{ (v, u) \;\middle|\; N \in \boldsymbol{\mathcal{N}}_{\mathcal{G},Y}, (v, u \in N \cap V(A)), v \neq u \right\}$$

Note the definition of Γ in Definition 20 is more general than the current problem requires as it returns a chordal graph over a general subset of $\boldsymbol{X}$, $A \subseteq \boldsymbol{X}$ instead of just over $\boldsymbol{Z}$. This general definition for Γ will be useful in later chapters, specifically Chapter 5. But, for the remainder of this chapter we will generally use $\boldsymbol{Z} = \boldsymbol{X} - \boldsymbol{Y}$ as the set of variables A that we want the graph returned by $\boldsymbol{\mathcal{N}}$-graph to have. In other words, for some $\boldsymbol{\mathcal{N}}$-partition $\boldsymbol{\mathcal{N}}_{\mathcal{G},Y}$, for the remainder of this chapter we will

call Γ with the arguments $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})$. Furthermore the graphs returned by Γ are chordal graphs as we will prove in Proposition 5.

Proposition 5 (The $\mathcal{N}$-graph created by Γ is a chordal graph) Let $\mathcal{G}$ be a chordal graph, $\boldsymbol{Y} \subset \boldsymbol{X}$, and $A \subseteq \boldsymbol{X}$. Then the $\boldsymbol{\mathcal{N}}$-graph, $\Gamma(A, \boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})$, is a chordal graph.

Proof We know that $\boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}}$ partitions the maximal cliques in $\boldsymbol{C}(\mathcal{G})$ such that each partition is a subtree of the junction tree/forest of $\mathcal{G}$. Therefore, by creating a graph $\mathcal{G}'$ where the vertices in each set in $\boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}}$ are fully connected to each other, the junction tree/forest of $\mathcal{G}'$ is equivalent to the junction tree/forest of $\mathcal{G}$ but with these subtrees merged together to form a single "super-clique" per partition $N \in \boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}}$. The resulting "super-cliques" will have the same parent as the root of the subtree and the same children as the leaves of the subtree. Therefore, the junction tree/forest of $\mathcal{G}'$ is valid which implies that $\mathcal{G}'$ is a chordal graph.

In order to obtain $\Gamma(A, \boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})$ from chordal graph $\mathcal{G}'$, all we need to do is delete the vertices that are *not* associated with the variables in A. From Corollary 1, we know that vertex deletion from a chordal graph preserves the chordal property of the graph. Therefore, $\Gamma(A, \boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})$ is also a chordal graph.

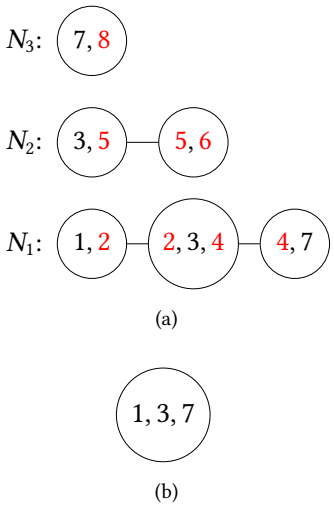


Figure 4.2: (a) An $\boldsymbol{\mathcal{N}}$-partition, $\boldsymbol{\mathcal{N}}$, and (b) the $\boldsymbol{\mathcal{N}}$-graph of $\boldsymbol{\mathcal{N}}$, $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}})$.

Example 3 Recall the junction tree, and specifically the $\boldsymbol{\mathcal{N}}$-partition $\boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}}$ from Example 1. See Figure 4.2a for an illustration of this $\boldsymbol{\mathcal{N}}$-partition from Example 1. Observe that $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})$, in Figure 4.2b, ensures that the variables $\boldsymbol{Y}$, highlighted in red, do not appear in the returned graph.

Due to this, the variables in $N_2 - \boldsymbol{Y}$ and $N_3 - \boldsymbol{Y}$ do not appear anywhere in $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})$ as a separate maximal clique. In other words, the number of maximal cliques in $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})$ is less than the number of sets in the partition $\boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}}$

$$\left|\boldsymbol{C}(\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}}))\right| \leq \left|\boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}}\right|$$

This also indicates that the marginal factors associated with variables in N_2 and N_3, $\varphi_{\{3,5,6\}-\{5,6\}}$ and $\varphi_{\{7,8\}-\{8\}}$ respectively, are either scalar factors, which is not the case here in this example, or have domains that are subsets of some existing clique in $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})$. Basic observation will show that the latter is the case in this example. Specifically, marginal factor $\varphi_{\{3,5,6\}-\{5,6\}}$ has a domain of $\boldsymbol{X}_3$ and $\varphi_{\{7,8\}-\{8\}}$ is defined over variable $\boldsymbol{X}_7$, both of which are subsets of $\boldsymbol{X}_{1,3,7}$.

We know that in general, for any $\boldsymbol{\mathcal{N}}$-partition $\boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}}$, the set of factors $\boldsymbol{\varphi}(\boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})$ has domains that are either empty, implying a scalar factor with all of the variables in

the factor before marginalisation being in $\boldsymbol{Y}$, or are equal to some, possibly *non-maximal*, clique in the graph $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})$. We know this must be the case because both

1. the domains of any non-scalar factors in $\boldsymbol{\varphi}(\boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})$, and
2. all the cliques, including *non-maximal* cliques, that are in $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})$,

are the same as they are constructed by taking each set in the partition $\boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}}$ and removing any vertices associated with $\boldsymbol{Y}$.

Furthermore, since all non-scalar factors in $\boldsymbol{\varphi}(\boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})$ have the same domain as the variables associated to some, possibly *non-maximal*, clique in $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})$, the domain of these factors is also either equal to, or a subset of, the variables associated with some *maximal* clique in $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})$. We shall call this mapping from factors in $\boldsymbol{\varphi}(\boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})$ to maximal clique in $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})$, μ.

Definition 21 (Mapping from $\boldsymbol{\varphi}(\boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})$ to $\boldsymbol{C}(\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}}))$, $\boldsymbol{\mu}$) Let $\mathcal{G}$ be a chordal graph and $\boldsymbol{Y} \subset \boldsymbol{X}$.

$$\begin{aligned} \mu \;:\; & \boldsymbol{\varphi}(\boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}}) \to \boldsymbol{C}(\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})) \\ \text{s.t.} \quad & \forall \phi \in \boldsymbol{\varphi}(\boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}}) \;:\; \text{Dom}(\phi) \subseteq \mu(\phi) \end{aligned}$$

where we take the convention that the empty set, {}, is a subset of any non-empty set.

Definition 22 (Inverse of μ, M) Let μ be a mapping from $\boldsymbol{\varphi}(\boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})$ to $\boldsymbol{C}(\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}}))$. Then we shall define the function M as an "inverse" of μ:

$$\begin{aligned} M \;:\; & \boldsymbol{C}(\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})) \to \mathcal{P}(\boldsymbol{\varphi}(\boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})) \\ \text{s.t.} \quad & \forall C \in \boldsymbol{C}(\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})) \;:\; M(C) = \left\{ \phi \;\middle|\; \phi \in \boldsymbol{\varphi}(\boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}}), \mu(\phi) = C \right\} \end{aligned}$$

Now that we have established a chordal graph $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})$ and a set of marginal $\boldsymbol{\mathcal{N}}$-probabilities $\boldsymbol{\varphi}(\boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})$ that do not contain any variable or vertices in $\boldsymbol{Y}$, and a mapping μ from these factors to maximal cliques of this chordal graph, there is one last thing we need verify in order for the connections in chordal graph $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})$ to "agree" with the set of marginal $\boldsymbol{\mathcal{N}}$-probability $\boldsymbol{\varphi}(\boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})$. That is, we need to verify that the product of factors assigned to each maximal clique in $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}})$, $\otimes M(C)$ for all $C \in \boldsymbol{C}(\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},\boldsymbol{Y}}))$, has the exact same domain as the variables associated with the maximal clique itself.

Proposition 6 (Mapping marginal factors to the marginal $\mathcal{N}$-graph) Let $\mathcal{G}$ be a choral graph, $Y \subset X$ the variable we wish to eliminate. Then there exists a mapping μ from the marginal factors $\boldsymbol{\varphi}(\boldsymbol{\mathcal{N}}_{\mathcal{G},Y})$ to the maximal cliques in the marginal $\boldsymbol{\mathcal{N}}_{\mathcal{G},Y}$-graph $\boldsymbol{C}(\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y}))$ such that the resulting product of factors assigned to each maximal clique is defined over the same variables as the variables associated with the respective clique.

Proof Recall that all the, possibly *non-maximal*, cliques in $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y})$ and all the factors in $\boldsymbol{\varphi}(\boldsymbol{\mathcal{N}}_{\mathcal{G},Y})$ are constructed by taking a set from the partition $\boldsymbol{\mathcal{N}}_{\mathcal{G},Y}$ and removing any vertices/variables in Y. Therefore, for all $C \in \boldsymbol{C}(\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y}))$ there exists a factor in $\boldsymbol{\varphi}(\boldsymbol{\mathcal{N}}_{\mathcal{G},Y})$ with variables exactly equal to the variables associated with the clique C.

$$\forall C \in \boldsymbol{C}(\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y}))\, \exists \phi \in \boldsymbol{\varphi}(\boldsymbol{\mathcal{N}}_{\mathcal{G},Y}) \,:\, \text{Vars}(\phi) = \boldsymbol{X}_C$$

This then implies that for any possible assignment μ, the resulting product factors in each maximal clique of $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y})$ will have the same variables as the variables associated with that maximal clique.

Therefore, we now have a set of factors $\boldsymbol{\varphi}(\boldsymbol{\mathcal{N}}_{\mathcal{G},Y})$ and a mapping μ from these factors to maximal cliques of chordal graph $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y})$, resulting in the set of factors $\Phi_{\mathcal{G}}$ defined over the maximal cliques of $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y})$:

$$\Phi_{\mathcal{G}} := \left\{ \prod_{\varphi \in M(C)} \varphi[\mathbb{D}_X] \;\middle|\; C \in \boldsymbol{C}(\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y})) \right\}$$

where $\forall C \in \boldsymbol{C}\left(\Gamma\left(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y}\right)\right)$

$$\phi_{\mathcal{G},C} := \prod_{\varphi \in M(C)} \varphi[\mathbb{D}_X]$$

From this, we know that

$$\forall C \in \boldsymbol{C}(\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y})) \,:\, \text{Vars}(\phi_{\mathcal{G},C}) = \boldsymbol{X}_C$$

and

$$\prod_{N \in \boldsymbol{\mathcal{N}}_{\mathcal{G},Y}} \varphi_{N-Y}[\mathbb{D}_X] = \prod_{C \in \boldsymbol{C}(\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y}))} \phi_{\mathcal{G},C}$$

since

$$\boldsymbol{\varphi}(\boldsymbol{\mathcal{N}}_{\mathcal{G},Y}) = \bigcup_{N \in \boldsymbol{\mathcal{N}}_{\mathcal{G},Y}} \varphi_{N-Y} = \bigcup_{C \in \boldsymbol{C}(\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y}))} \bigcup_{\varphi \in M(C)} \varphi$$

Therefore, we have reframed the marginal distribution over variables $Z \subset X, Y = X - Z$, $\mathbb{P}_Z$, of a DM $\mathbb{D}_\mathcal{G}$ as the un-normalised *joint* distribution of a new MN $\mathbb{D}_{\mathcal{G}'}$, where $\mathcal{G}'$ is the chordal graph $\Gamma(Z, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y})$, with the set of marginal $\boldsymbol{\mathcal{N}}$-probabilities as factors of the new MN.

$$\mathbb{D}_Z = \prod_{N \in \boldsymbol{\mathcal{N}}_{\mathbb{D},Y}} \varphi_{N-Y}[\mathbb{D}] = \prod_{C \in \boldsymbol{C}(\Gamma(Z, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y}))} \phi_{\mathcal{G},C}$$

Then, as we will describe in the next section, we can use this transformation to reframe the problem of computing the $\alpha\beta$-divergence between the marginal distributions of two DMs into a form that can be computed using the methods developed in Chapter 3.

4.3 Computing $\alpha\beta$-Divergence

Instead of tackling the problem of computing the $\alpha\beta$-divergence between the marginal distributions of two DMs, $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$ and $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$, directly, we will reframe this problem as computing the $\alpha\beta$-divergence between the un-normalised *joint* distribution of two new chordal MNs, $\mathbb{P}_{\mathcal{G}'_\mathbb{P}}$ and $\mathbb{Q}_{\mathcal{G}'_\mathbb{Q}}$, by using the transformations described in Section 4.2 on DMs $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$ and $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$ respectively. Then, since we know how to compute the $\alpha\beta$-divergence between the joint distribution of two DMs, i.e. two chordal MNs, we can use the methods described in Chapter 3 to compute the $\alpha\beta$-divergence between $\mathbb{P}_{\mathcal{G}'_\mathbb{P}}$ and $\mathbb{Q}_{\mathcal{G}'_\mathbb{Q}}$.

First, let us establish the transformations for the original DMs, $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$ and $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$, that we will use in this section. For some marginal variable set $Z \subset X$ where $Y = X - Z$, and $\boldsymbol{\mathcal{N}}$-partition $\boldsymbol{\mathcal{N}}_{\mathcal{G}_\mathbb{P},Y}$ and $\boldsymbol{\mathcal{N}}_{\mathcal{G}_\mathbb{P},Y}$ for DMs $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$ and $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$ respectively, we shall define the following chordal graphs and factor sets using the transformations laid out in Section 4.2:

$$\begin{aligned}
\mathcal{G}'_\mathbb{P} &:= \Gamma\left(Z, \boldsymbol{\mathcal{N}}_{\mathcal{G}_\mathbb{P},Y}\right) \\
\Phi_\mathbb{P} &:= \left\{ \prod_{\varphi \in M(C)} \varphi[\mathbb{D}_X] \;\middle|\; C \in \boldsymbol{C}\left(\Gamma\left(Z, \boldsymbol{\mathcal{N}}_{\mathcal{G}_\mathbb{P},Y}\right)\right) \right\} \\
\mathcal{G}'_\mathbb{Q} &:= \Gamma\left(Z, \boldsymbol{\mathcal{N}}_{\mathcal{G}_\mathbb{Q},Y}\right) \\
\Phi_\mathbb{Q} &:= \left\{ \prod_{\varphi \in M(C)} \varphi[\mathbb{D}_X] \;\middle|\; C \in \boldsymbol{C}\left(\Gamma\left(Z, \boldsymbol{\mathcal{N}}_{\mathcal{G}_\mathbb{Q},Y}\right)\right) \right\}
\end{aligned}$$

and as we have shown in Section 4.2, this transformation allows us to express the marginal probabilities $\mathbb{P}_Z$ and $\mathbb{Q}_Z$ in terms of a product of factors over the maximal

cliques of $\Gamma\left(Z, \mathcal{N}_{\mathcal{G}_{\mathbb{P}},Y}\right)$ and $\Gamma\left(Z, \mathcal{N}_{\mathcal{G}_{\mathbb{Q}},Y}\right)$ respectively.

$$\mathbb{P}_Z = \prod_{C \in \boldsymbol{C}(\Gamma(Z,\mathcal{N}_{\mathbb{P},Y}))} \phi_{\mathbb{P},C} \qquad \mathbb{Q}_Z = \prod_{C \in \boldsymbol{C}(\Gamma(Z,\mathcal{N}_{\mathbb{Q},Y}))} \phi_{\mathbb{Q},C}$$

Then recall the extended $\alpha\beta$-divergence from Definition 11 and that its expression depends on the value of the parameters α and β:

$$D_{AB}^{\alpha,\beta}(\mathbb{P},\mathbb{Q}) = \sum_{\boldsymbol{x}\in\mathcal{X}} d_{AB}^{\alpha,\beta}\big(\mathbb{P}(\boldsymbol{x}),\mathbb{Q}(\boldsymbol{x})\big)$$

where

$$d_{AB}^{(\alpha,\beta)}(\mathbb{P}(\boldsymbol{x}),\mathbb{Q}(\boldsymbol{x})) = \begin{cases} -\frac{1}{\alpha\beta}\left(\mathbb{P}(\boldsymbol{x})^\alpha\mathbb{Q}(\boldsymbol{x})^\beta - \frac{\alpha\mathbb{P}(\boldsymbol{x})^{\alpha+\beta}}{\alpha+\beta} - \frac{\beta\mathbb{Q}(\boldsymbol{x})^{\alpha+\beta}}{\alpha+\beta}\right) & \text{for } \alpha,\beta,\alpha+\beta\neq 0 \\ \frac{1}{\alpha^2}\left(\mathbb{P}(\boldsymbol{x})^\alpha \log\frac{\mathbb{P}(\boldsymbol{x})^\alpha}{\mathbb{Q}(\boldsymbol{x})^\alpha} - \mathbb{P}(\boldsymbol{x})^\alpha + \mathbb{Q}(\boldsymbol{x})^\alpha\right) & \text{for } \alpha\neq 0,\beta=0 \\ \frac{1}{\alpha^2}\left(\log\frac{\mathbb{Q}(\boldsymbol{x})^\alpha}{\mathbb{P}(\boldsymbol{x})^\alpha} + \left(\frac{\mathbb{Q}(\boldsymbol{x})^\alpha}{\mathbb{P}(\boldsymbol{x})^\alpha}\right)^{-1} - 1\right) & \text{for } \alpha=-\beta\neq 0 \\ \frac{1}{\beta^2}\left(\mathbb{Q}(\boldsymbol{x})^\beta \log\frac{\mathbb{Q}(\boldsymbol{x})^\beta}{\mathbb{P}(\boldsymbol{x})^\beta} - \mathbb{Q}(\boldsymbol{x})^\beta + \mathbb{P}(\boldsymbol{x})^\beta\right) & \text{for } \alpha=0,\beta\neq 0 \\ \frac{1}{2}(\log\mathbb{P}(\boldsymbol{x}) - \log\mathbb{Q}(\boldsymbol{x}))^2 & \text{for } \alpha,\beta=0. \end{cases}$$

and that in cases *other* than $\alpha,\beta = 0$, the $\alpha\beta$-divergence can be expressed as a linear combination of the functional $\mathcal{F}$ from Definition 13

$$\mathcal{F}(\mathbb{P},\mathbb{Q};g,h,g^*,h^*) = \sum_{\boldsymbol{x}\in\mathcal{X}} \Big(g\big[\mathbb{P}\big](\boldsymbol{x})\Big)\Big(h\big[\mathbb{Q}\big](\boldsymbol{x})\Big) L\Big(\Big(g^*[\mathbb{P}](\boldsymbol{x})\Big)\Big(h^*[\mathbb{Q}](\boldsymbol{x})\Big)\Big)$$

as we have shown in Theorem 7. Therefore, we will first tackle computing the $\alpha\beta$-divergence when $\alpha,\beta = 0$ before tackling computing $\mathcal{F}$ which will cover computing the $\alpha\beta$-divergence for other values of α and β.

4.3.1 Computing Marginal $\alpha\beta$-Divergence when $\alpha,\beta = 0$

We can compute the $\alpha\beta$-divergence when $\alpha,\beta = 0$ between the marginal distributions of $\mathbb{P}_{\mathcal{G}_{\mathbb{P}}}$ and $\mathbb{Q}_{\mathcal{G}_{\mathbb{Q}}}$, $D_{AB}^{(0,0)}(\mathbb{P}_Z,\mathbb{Q}_Z)$, by substituting the alternate expressions for $\mathbb{P}_Z$ and $\mathbb{Q}_Z$ in Section 4.3.

$$\begin{aligned} D_{AB}^{(0,0)}(\mathbb{P}_Z,\mathbb{Q}_Z) &= \sum_{z\in\mathcal{Z}} \frac{1}{2}\left(\log\mathbb{P}_Z(z) - \log\mathbb{Q}_Z(z)\right)^2 \\ &= \sum_{z\in\mathcal{Z}} \frac{1}{2}\left(\log\left(\prod_{C\in\boldsymbol{C}(\Gamma(Z,\mathcal{N}_{\mathbb{P},Y}))}\phi_{\mathbb{P},C}\right)(z) - \log\left(\prod_{C\in\boldsymbol{C}(\Gamma(Z,\mathcal{N}_{\mathbb{Q},Y}))}\phi_{\mathbb{Q},C}\right)(z)\right)^2 \end{aligned}$$

Therefore, the functional $D_{AB}^{(0,0)}(\mathbb{P}_Z, \mathbb{Q}_Z)$ is equivalent to the same functional between the product of factors over the maximal cliques of $\Gamma\left(Z, \boldsymbol{\mathcal{N}}_{\mathcal{G}_\mathbb{P},Y}\right)$ and $\Gamma\left(Z, \boldsymbol{\mathcal{N}}_{\mathcal{G}_\mathbb{Q},Y}\right)$ respectively. As a result, we can use the argument in Theorem 8 to show that $D_{AB}^{(0,0)}(\mathbb{P}_Z, \mathbb{Q}_Z)$ can be computed in time exponential to the maximum clique size of graphs $\Gamma(Z, \boldsymbol{\mathcal{N}}_{\mathbb{P},Y})$ and $\Gamma(Z, \boldsymbol{\mathcal{N}}_{\mathbb{P},Y})$.

4.3.2 Computing $\mathcal{F}$ between Marginal Distributions of Directed Graphs

In order to compute the functional $\mathcal{F}$ between the marginal distributions of DMs $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$ and $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$, $\mathcal{F}(\mathbb{P}_Z, \mathbb{Q}_Z)$, we can plug in the alternate expressions for $\mathbb{P}_Z$ and $\mathbb{Q}_Z$ in Section 4.3 into $\mathcal{F}$.

$$
\begin{aligned}
&\mathcal{F}(\mathbb{P}_Z, \mathbb{Q}_Z) \\
&= \sum_{z \in \mathcal{Z}} g\left[\mathbb{P}_Z\right](z) \cdot h\left[\mathbb{Q}_Z\right](z) \cdot L\left(g^+\left[\mathbb{P}_Z\right](z) \cdot h^+\left[\mathbb{Q}_Z\right](z)\right) \\
&= \sum_{z \in \mathcal{Z}} g\left[\mathbb{P}_Z\right](z) \cdot h\left[\mathbb{Q}_Z\right](z) \cdot L\left(g^+\left[\prod_{C \in \boldsymbol{C}(\mathcal{G}'_\mathbb{P})} \phi_{\mathbb{P},C}\right](z) \cdot h^+\left[\prod_{C \in \boldsymbol{C}(\mathcal{G}'_\mathbb{Q})} \phi_{\mathbb{Q},C}\right](z)\right) \\
&= \sum_{z \in \mathcal{Z}} g\left[\mathbb{P}_Z\right](z) \cdot h\left[\mathbb{Q}_Z\right](z) \cdot L\left(\left(\prod_{C \in \boldsymbol{C}(\mathcal{G}'_\mathbb{P})} g^+\left[\phi_{\mathbb{P},C}\right](z)\right) \cdot \left(\prod_{C \in \boldsymbol{C}(\mathcal{G}'_\mathbb{Q})} h^+\left[\phi_{\mathbb{Q},C}\right](z)\right)\right) \\
&= \sum_{C \in \boldsymbol{C}(\mathcal{G}'_\mathbb{P})} \sum_{z_C \in \mathcal{Z}_C} L\left(g^+\left[\phi_{\mathbb{P},C}\right](z)\right) SP_C(z_C) + \sum_{C \in \boldsymbol{C}(\mathcal{G}'_\mathbb{Q})} \sum_{z_C \in \mathcal{Z}_C} L\left(h^+\left[\phi_{\mathbb{Q},C}\right](z)\right) SP_C(z_C)
\end{aligned}
\tag{4.6}
$$

where

$$
\begin{aligned}
\mathcal{G}'_\mathbb{P} &:= \Gamma(Z, \boldsymbol{\mathcal{N}}_{\mathbb{P},Y}) \\
\mathcal{G}'_\mathbb{Q} &:= \Gamma(Z, \boldsymbol{\mathcal{N}}_{\mathbb{Q},Y}) \\
SP_C(z_C) &= \sum_{z \in \mathcal{Z}_{Z-C}} \left(\prod_{C \in \boldsymbol{C}(\mathcal{G}'_\mathbb{P})} g\left[\phi_{\mathbb{P},C}\right](z_C, z)\right) \cdot \left(\prod_{C \in \boldsymbol{C}(\mathcal{G}'_\mathbb{Q})} h\left[\phi_{\mathbb{Q},C}\right](z_C, z)\right)
\end{aligned}
$$

We can then obtain SP_C with MGASP outlined in Section 3.2 by using the chordal graphs $\mathcal{G}'_\mathbb{P}$ and $\mathcal{G}'_\mathbb{Q}$ with factors

$$
\begin{aligned}
\Psi_\mathbb{P} &:= \left\{ g\left[\phi_{\mathbb{P},C}\right] \;\middle|\; C \in \Gamma(Z, \boldsymbol{\mathcal{N}}_{\mathbb{P},Y}) \right\} \\
\Psi_\mathbb{Q} &:= \left\{ h\left[\phi_{\mathbb{Q},C}\right] \;\middle|\; C \in \Gamma(Z, \boldsymbol{\mathcal{N}}_{\mathbb{Q},Y}) \right\}
\end{aligned}
$$

respectively. Once we have obtained all the needed SP_C, we can continue with the computation of $\mathcal{F}(\mathbb{P}_Z, \mathbb{Q}_Z)$ by directly computing the nested sums in Equation (4.6). Since computing $\mathcal{F}(\mathbb{P}_Z, \mathbb{Q}_Z)$ is essentially similar to computing $\mathcal{F}(\mathbb{P}, \mathbb{Q})$ but with the new pair of chordal graphs and factor sets, $(\mathcal{G}'_{\mathbb{P}}, \Psi_{\mathbb{P}})$ and $(\mathcal{G}'_{\mathbb{Q}}, \Psi_{\mathbb{P}})$, the complexity of computing $\mathcal{F}(\mathbb{P}_Z, \mathbb{Q}_Z)$ is essentially the same as the complexity of $\mathcal{F}(\mathbb{P}, \mathbb{Q})$. The main difference in complexity is that computing $\mathcal{F}(\mathbb{P}_Z, \mathbb{Q}_Z)$ scales exponentially in the treewidth of the computation graph of $\mathcal{G}'_{\mathbb{P}}$ and $\mathcal{G}'_{\mathbb{Q}}$ instead of the computation graph of $\mathcal{G}_{\mathbb{P}}$ and $\mathcal{G}_{\mathbb{Q}}$.

4.4 Complexity and Edge Cases

In this section, we will discuss the computational complexity of using the transformations described in this chapter in order to compute the $\alpha\beta$-divergence between the marginal distributions of two DMs. Furthermore, we will show, via two examples, that the computational complexity of computing this marginal $\alpha\beta$-divergence varies greatly depending on the choice of marginal variables $\boldsymbol{Z} \subset \boldsymbol{X}$, ranging from being exponential in the treewidth of the original DMs, to exponential in the total number of variables in the original DMs, $|\boldsymbol{X}|$.

Recall that the complexity of computing $\alpha\beta$-divergence between joint distributions of DMs is

$$D_{AB}^{(\alpha,\beta)}(\mathbb{P}\|\mathbb{Q}) \in \begin{cases} \mathcal{O}(|\boldsymbol{X}|^2 \cdot \omega_{\max} 2^{\omega_{\max}+1}) & \alpha, \beta = 0 \\ \mathcal{O}(|\boldsymbol{X}| \cdot 2^{\omega(\mathcal{H})+1}) & \text{otherwise} \end{cases} \tag{3.10}$$

where

$$\omega_{\max} = \max(\omega(\mathcal{G}_{\mathbb{P}}), \omega(\mathcal{G}_{\mathbb{Q}}))$$

We have shown in Section 4.3, that the problem of computing the $\alpha\beta$-divergence between marginal distributions of DMs can be reframed as computing the $\alpha\beta$-divergence between the product of factors over the maximal cliques of the pair of new chordal graphs,

$$\begin{aligned} \mathcal{G}'_{\mathbb{P}} &:= \Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathbb{P},Y}) \\ \mathcal{G}'_{\mathbb{Q}} &:= \Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathbb{Q},Y}) \end{aligned}$$

respectively. Therefore, assuming $\mathcal{H}'$ is the computation graph between $\mathcal{G}'_{\mathbb{P}}$ and $\mathcal{G}'_{\mathbb{Q}}$, the complexity of computing the $\alpha\beta$-divergence between marginal distributions of two DMs is

$$D_{AB}^{(\alpha,\beta)}(\mathbb{P}_Z\|\mathbb{Q}_Z) \in \begin{cases} \mathcal{O}(|\boldsymbol{X}|^2 \cdot \omega'_{\max} 2^{\omega'_{\max}+1}) & \alpha, \beta = 0 \\ \mathcal{O}(|\boldsymbol{X}| \cdot 2^{\omega(\mathcal{H}')+1}) & \text{otherwise} \end{cases}$$

where

$$\omega'_{\max} = \max(\omega(\mathcal{G}'_{\mathbb{P}}), \omega(\mathcal{G}'_{\mathbb{Q}}))$$

It is tempting to end the discussion regarding the computational complexity of computing $D_{AB}^{(\alpha,\beta)}(\mathbb{P}_Z, \mathbb{Q}_Z)$ here. However, we still need to take into account the computational complexity of obtaining the set of marginal $\boldsymbol{\mathcal{N}}$-probabilities $\boldsymbol{\varphi}\left(\boldsymbol{\mathcal{N}}_{\mathbb{P},Y}\right)$ and $\boldsymbol{\varphi}\left(\boldsymbol{\mathcal{N}}_{\mathbb{Q},Y}\right)$ for marginal probabilities $\mathbb{P}_Z$ and $\mathbb{Q}_Z$ respectively.

Recall from Proposition 4 that the complexity of obtaining the marginal $\boldsymbol{\mathcal{N}}$-probability for some set N in a $\boldsymbol{\mathcal{N}}$-partition is

$$\mathcal{O}(2^{|N|})$$

Then the complexity of obtaining the set of marginal $\boldsymbol{\mathcal{N}}$-probabilities for some $\boldsymbol{\mathcal{N}}$-partition $\boldsymbol{\mathcal{N}}_{\mathbb{D},Y}$ is

$$\sum_{N\in\boldsymbol{\mathcal{N}}_{\mathbb{D},Y}} \mathcal{O}(2^{|N|}) \in \max_{N\in\boldsymbol{\mathcal{N}}_{\mathbb{D},Y}} \mathcal{O}(\left|\boldsymbol{\mathcal{N}}_{\mathbb{D},Y}\right| \cdot 2^{|N|}) \in \max_{N\in\boldsymbol{\mathcal{N}}_{\mathbb{D},Y}} \mathcal{O}(|\boldsymbol{X}| \cdot 2^{|N|})$$

and the total complexity of obtaining the marginal $\boldsymbol{\mathcal{N}}$-probabilities for both $\boldsymbol{\mathcal{N}}_{\mathbb{P},Y}$ and $\boldsymbol{\mathcal{N}}_{\mathbb{Q},Y}$ is just

$$\max_{\boldsymbol{\mathcal{N}}\in(\boldsymbol{\mathcal{N}}_{\mathbb{P},Y},\boldsymbol{\mathcal{N}}_{\mathbb{Q},Y})} \max_{N\in\boldsymbol{\mathcal{N}}} \mathcal{O}(|\boldsymbol{X}| \cdot 2^{|N|}) \in \mathcal{O}(|\boldsymbol{X}| \cdot 2^{\nu})$$

where

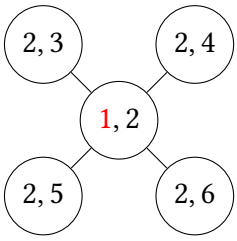
$$\nu := \max_{\boldsymbol{\mathcal{N}}\in(\boldsymbol{\mathcal{N}}_{\mathbb{P},Y},\boldsymbol{\mathcal{N}}_{\mathbb{Q},Y})} \max_{N\in\boldsymbol{\mathcal{N}}} |N|$$

Therefore, the final complexity of computing $D_{AB}^{(\alpha,\beta)}(\mathbb{P}_Z, \mathbb{Q}_Z)$ is

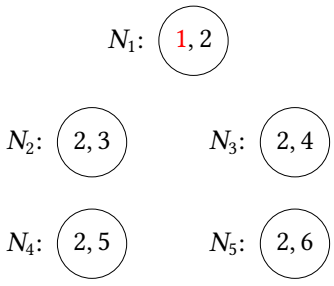
$$\mathcal{O}(|\boldsymbol{X}| \cdot 2^{\nu}) + \begin{cases} \mathcal{O}(|\boldsymbol{X}|^2 \cdot \omega'_{\max} 2^{\omega'_{\max}+1}) & \alpha, \beta = 0 \\ \mathcal{O}(|\boldsymbol{X}| \cdot 2^{\omega(\mathcal{H}')+1}) & \text{otherwise} \end{cases}$$

since the computational complexity of $D_{AB}^{(\alpha,\beta)}(\mathbb{P}_Z, \mathbb{Q}_Z)$ depends on the size of the largest vertex set in the $\boldsymbol{\mathcal{N}}$-partitions $\boldsymbol{\mathcal{N}}_{\mathbb{P},Y}$ and $\boldsymbol{\mathcal{N}}_{\mathbb{Q},Y}$. This in turn, is *highly dependent* on the set of variables, Y, we wish to marginalise out. As we will demonstrate in the following examples, it is possible for the computational complexity of $D_{AB}^{(\alpha,\beta)}(\mathbb{P}_Z, \mathbb{Q}_Z)$ to be as varied as having the same complexity as computing the $\alpha\beta$-divergence between joint distributions of $\mathbb{P}_{\mathcal{G}_{\mathbb{P}}}$ and $\mathbb{Q}_{\mathcal{G}_{\mathbb{Q}}}$, to having a complexity that is exponential with respect to the total number of variables in the DMs $|\boldsymbol{X}|$.

Example 4 Consider a chordal graph $\mathcal{G}$ with the junction tree in Figure 4.3a. Furthermore, assume we want to marginalise out the variable $Y = \{X_1\}$. Then



(a) Clique tree of a chordal graph.



(b) Junction trees of the sets in one possible $\boldsymbol{\mathcal{N}}$-partition for variables of interest.

Figure 4.3: A clique tree and possible $\boldsymbol{\mathcal{N}}$-partitions for the variables highlighted in red.

one possible $\mathcal{N}$-partition for the graph $\mathcal{G}$ and variables Y, $\mathcal{N}_{\mathcal{G},Y}$, can be found in Figure 4.3b.

Observe that the sets in $\mathcal{N}_{\mathcal{G},Y}$ are just the individual maximal cliques of $\mathcal{G}$. This is because the variable we wish to marginalise out, Y, is not contained in any minimal separator in the junction tree of $\mathcal{G}$. This results in the treewidth of both $\mathcal{G}$ and $\Gamma(Z, \mathcal{N}_{\mathcal{G},Y})$ to be the same.

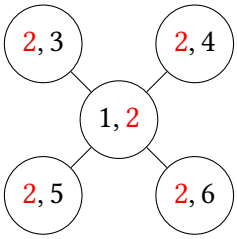
Therefore, assuming we have two DMs with the same chordal graph structure $\mathcal{G}$, $\mathbb{P}_{\mathcal{G}}$ and $\mathbb{Q}_{\mathcal{G}}$, the treewidth of $\mathcal{G}$ will be the same as the treewidth of $\Gamma(Z, \mathcal{N}_{\mathcal{G},Y})$.

$$\omega(\mathcal{G}) = \omega(\Gamma(Z, \mathcal{N}_{\mathcal{G},Y})) = 1$$

Furthermore, the size of the largest set in the $\mathcal{N}$-partition $\mathcal{N}_{\mathcal{G},Y}$ is $\nu = \omega(\mathcal{G}) + 1 = 2$ as well. Additionally, the computation graph between both DMs and also both $\mathcal{N}$-graphs is just the graph $\mathcal{G}$. Therefore the complexity of computing $D_{AB}^{(\alpha,\beta)}(\mathbb{P}_Z, \mathbb{Q}_Z)$ in this example is:

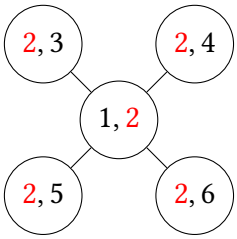
$$\mathcal{O}(|X| \cdot 2^2) + \begin{cases} \mathcal{O}(|X|^2 \cdot 2^2) & \alpha, \beta = 0 \\ \mathcal{O}(|X| \cdot 2^2) & \text{otherwise} \end{cases} \in \begin{cases} \mathcal{O}(|X|^2 \cdot 2^2) & \alpha, \beta = 0 \\ \mathcal{O}(|X| \cdot 2^2) & \text{otherwise} \end{cases}$$

which is the same complexity of computing the joint distributions of DMs $\mathbb{P}_{\mathcal{G}}$ and $\mathbb{P}_{\mathcal{G}}$ in this example.



(a) Clique tree of a chordal graph.

N_1:



(b) Junction tree of the only possible $\mathcal{N}$-partition for variables of interest.

Figure 4.4: A clique tree and possible $\mathcal{N}$-partitions for the variables highlighted in red.

Example 5 Consider the chordal graph $\mathcal{G}$ with the junction tree in Figure 4.4a. Furthermore, assume we want to marginalise out the variable $Y = \{X_2\}$. Then the *only* possible $\mathcal{N}$-partition for the graph $\mathcal{G}$ and variables Y, $\mathcal{N}_{\mathcal{G},Y}$, is the partition containing a single vertex set, with the set itself containing all the vertices in $\mathcal{G}$.

Therefore the treewidth of the $\mathcal{N}$-graph $\Gamma(Z, \mathcal{N}_{\mathcal{G},Y})$ is just 4, i.e. the number of variables in $Z = X - \{X_2\}$ minus one, and the size of the "largest" set in $\mathcal{N}_{\mathcal{G},Y}$ is just the number of variables $|X| = 6$.

Since, for two DMs with the graph structure $\mathcal{G}$, $\mathbb{P}_{\mathcal{G}}$ and $\mathbb{Q}_{\mathcal{G}}$, their $\mathcal{N}$-partition, and therefore, $\mathcal{N}$-graph is the same, the complexity of computing the $\alpha\beta$-divergence between the marginal distribution over Z of DMs $\mathbb{P}_{\mathcal{G}}$ and $\mathbb{Q}_{\mathcal{G}}$ is

$$\mathcal{O}(|X| \cdot 2^{|X|}) + \begin{cases} \mathcal{O}(|X|^2(|X| - 2) \cdot 2^{|X|-1}) & \alpha, \beta = 0 \\ \mathcal{O}(|X| \cdot 2^{|X|-1}) & \text{otherwise} \end{cases}$$

which implies that the complexity of computing $D_{AB}^{(\alpha,\beta)}(\mathbb{P}_Z, \mathbb{Q}_Z)$ is at least exponential with respect to the number of variables in X, $|X|$.

4.5 Conclusion

To summarise, in this chapter we have shown how to decompose the marginal distribution $\mathbb{D}_Z, \boldsymbol{Z} \subset \boldsymbol{X}$ of a DM $\mathbb{D}_\mathcal{G}$ and how to use this decomposition to compute the $\alpha\beta$-divergence between marginal distributions of DMs.

This decomposition involves first creating a partition of the maximal cliques of the chordal graph $\mathcal{G}$ and the vertices associated with the variables we wish to marginalise out, $\boldsymbol{Y} = \boldsymbol{X} - \boldsymbol{Z}$. We call such partitions $\boldsymbol{\mathcal{N}}$-partitions if and only if, for all sets in the $\boldsymbol{\mathcal{N}}$-partition, $N \in \boldsymbol{\mathcal{N}}_{\mathcal{G},Y}$, the junction tree of the induced subgraph $\mathcal{G}(N)$ is a subtree of the junction tree of $\mathcal{G}$ itself (see Definition 18). Because $\boldsymbol{\mathcal{N}}_{\mathcal{G},Y}$ partitions the maximal cliques in $\mathcal{G}$, there is a many-to-one mapping from the CPT of each maximal clique in $\mathbb{D}_\mathcal{G}$ to a single set in the $\boldsymbol{\mathcal{N}}$-partition $\boldsymbol{\mathcal{N}}_{\mathcal{G},Y}$. We can then even take the product of the CPTs assigned to each set $N \in \boldsymbol{\mathcal{N}}_{\mathcal{G},Y}$ and marginalise out any variables in $\boldsymbol{Y}$ to obtain the set of marginal $\boldsymbol{\mathcal{N}}$-probabilities $\boldsymbol{\varphi}(\boldsymbol{\mathcal{N}}_{\mathcal{G},Y})$ (see Definition 19). We then show that the product of the marginal $\boldsymbol{\mathcal{N}}$-probabilities in $\boldsymbol{\varphi}(\boldsymbol{\mathcal{N}}_{\mathcal{G},Y})$ is equivalent to the marginal probability $\mathbb{D}_Z$, thus providing the decomposition of $\mathbb{D}_Z$ that we are looking for.

In order to use this decomposition of $\mathbb{D}_Z$ for some DM $\mathbb{D}_\mathcal{G}$ to assist in computing the $\alpha\beta$-divergence between marginal distributions of two DMs, we need to re-express this decomposition of $\mathbb{D}_Z$ as a product of factors defined over the maximal cliques of some chordal graph. This is so that we can reuse the methods established in Chapter 3 for computing the $\alpha\beta$-divergence between marginal distributions.

Therefore, the first step we took to find this re-expression of $\mathbb{D}_Z$ is to create a new chordal graph, that we call the $\boldsymbol{\mathcal{N}}$-graph and denote by $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y})$. We construct the $\boldsymbol{\mathcal{N}}$-graph by fully connecting vertices of each set in the $\boldsymbol{\mathcal{N}}$-partition $\boldsymbol{\mathcal{N}}_{\mathcal{G},Y}$. However, we also remove any vertices associated with the variables we wish to marginalise out, $\boldsymbol{Y}$, from $\boldsymbol{\mathcal{N}}_{\mathcal{G},Y}$ (see Definition 20). There is then a many-to-one mapping from the marginal $\boldsymbol{\mathcal{N}}$-probabilities in $\boldsymbol{\varphi}(\boldsymbol{\mathcal{N}}_{\mathcal{G},Y})$ to maximal cliques in the $\boldsymbol{\mathcal{N}}$-graph, $\boldsymbol{C}(\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y}))$ (see Definition 21). Therefore, we defined a new set of factors, $\Phi_\mathcal{G}$, for each maximal clique in $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y})$, where each factor in $\Phi_\mathcal{G}$ is a product of the marginal $\boldsymbol{\mathcal{N}}$-probabilities from $\boldsymbol{\varphi}(\boldsymbol{\mathcal{N}}_{\mathcal{G},Y})$ mapped to the factor's respective maximal clique. We then have a set of factors $\Phi_\mathcal{G}$ defined over the maximal cliques of chordal graph $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y})$, where the product of the factors in $\Phi_\mathcal{G}$ is also equivalent to the marginal distribution $\mathbb{D}_Z$. This is exactly the expression of $\mathbb{D}_Z$ we need in order to use $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y})$ and $\Phi_\mathcal{G}$ as inputs to the methods described in Chapter 3.

With all this machinery established, all we need to do to compute the $\alpha\beta$-divergence between the marginal distributions, $\mathbb{P}_Z$ and $\mathbb{Q}_Z$, of two DMs, $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$ and $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$, is to find their respective $\boldsymbol{\mathcal{N}}$-graphs, $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathbb{P},Y})$ and $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathbb{Q},Y})$, and factor sets, $\Phi_\mathbb{P}$ and $\Phi_\mathbb{Q}$. We can then directly use the methods from Chapter 3 to compute the

$\alpha\beta$-divergence between these $\mathcal{N}$-graphs and their factors. The result we obtain from this computation will be exactly the $\alpha\beta$-divergence between the marginal distributions $\mathbb{P}_Z$ and $\mathbb{Q}_Z$.

Unfortunately, as far as we are aware, there is no existing work on computing the divergence between the marginal distributions of two probabilistic graphical models. Furthermore, the runtime for our approach to computing the marginal divergence is theorectically similar to computing the joint divergence in Section 3.2, but with $\mathcal{N}$-graphs being our initial graphs instead. Therefore, we omit any runtime experiment in this chapter.

Chapter 5

Computing Divergences between the Conditional Distributions of 2 Decomposable Models

So far we have shown how to compute the $\alpha\beta$-divergence of both the joint and marginal distributions between two DMs. However, a core problem in machine learning is the prediction of the values over a set of variables, Y, given the values over another set of variables Z, i.e. a classification problem. In problems such as these, the main distribution of interest is the distribution of the variables Y conditioned on some value over the variables Z, i.e. the conditional distribution of Y given Z. Such systems may be impacted by drift in the conditional distribution differently, and often more severely, compared to drift in the joint distribution or marginal distributions.

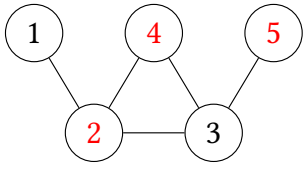
Therefore, in addition to the joint and marginal distributions, we require a method to measure the divergence between conditional distributions encoded within two DMs.

Recall the general conditional divergence chosen in Section 2.4.1:

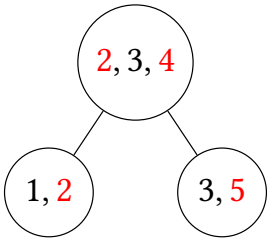
$$\begin{aligned} D(\mathbb{P}_{Y|Z} \parallel \mathbb{Q}_{Y|Z}) &= \mathbb{E}_{Z\sim\mathbb{P}}\left[D(\mathbb{P}_{Y|Z} \parallel \mathbb{Q}_{Y|Z})\right] \\ &= \sum_{z\in\mathcal{Z}} \mathbb{P}_Z(z) D(\mathbb{P}_{Y|Z=z} \parallel \mathbb{Q}_{Y|Z=z}) \end{aligned}$$

In order to efficiently compute the divergence between conditional distributions of two DMs, we need to find a decomposition of the distributions in the conditional divergence. We already know how to decompose the marginal distribution $\mathbb{P}_Z$ from Chapter 4, therefore, the problem we need to tackle in this chapter is finding a decomposition of the conditional distribution encoded in a DM.

In order to better illustrate this problem, let us consider Example 6 of finding a decomposition of the conditional distribution for the chordal graph in Figure 5.1a.



(a) Chordal graph.



(b) Junction tree of chordal graph.

Figure 5.1: (a) an example of a chordal graph and (b) its junction tree.

Example 6 Let $\mathbb{D}_{\mathcal{G}}$ be a decomposable model where $\mathcal{G}$ is a chordal graph with the following vertices and edges

$$V(\mathcal{G}) = \{1, 2, 3, 4, 5\}$$
$$E(\mathcal{G}) = \{(1, 2), (2, 3), (2, 4), (3, 4), (3, 5)\}$$

as illustrated in Figure 5.1a. $\mathcal{G}$ also has a junction tree as illustrated in Figure 5.1b. Furthermore, assume that we split the set of all variables in $\mathbb{D}_{\mathcal{G}}$, $\boldsymbol{X}$, into 2 mutually exclusive sets:

$$\boldsymbol{Z} = \{1, 3\}$$
$$\boldsymbol{Y} = \{2, 4, 5\}$$

Therefore, we now have the conditional distribution

$$\mathbb{D}_{Y|Z} = \mathbb{D}_{2,4,5|1,3}$$

that we wish to decompose into a product of smaller distributions.

Recall from Definition 4 that when two sets of variables $\boldsymbol{X}_A$ and $\boldsymbol{X}_B$ are conditionally independent given the rest of the variables $\boldsymbol{Z} = \boldsymbol{X} - (\boldsymbol{X}_A \cup \boldsymbol{X}B)$, we can decompose the conditional distribution of $\boldsymbol{X}_{A,B}$ given $\boldsymbol{Z}$ into the product of 2 conditional distributions:

$$\boldsymbol{X}_A \perp\!\!\!\perp \boldsymbol{X}_B \mid \boldsymbol{Z} \Leftrightarrow \mathbb{D}_{A,B|Z} = \mathbb{D}_{A|Z}\mathbb{D}_{B|Z}$$

Furthermore, recall from Definition 7 that $\boldsymbol{X}_A$ and $\boldsymbol{X}_B$ are conditionally independent given $\boldsymbol{Z}$ if and only if the vertices $V(\boldsymbol{Z})$ separate the vertices A and B in the graph containing these mutually exclusive set of vertices.

$$\boldsymbol{X}_A \perp\!\!\!\perp \boldsymbol{X}_B \mid \boldsymbol{Z} \Leftrightarrow \text{sep}_{\mathcal{G}}(A; B \mid V(\boldsymbol{Z}))$$

Specifically, in our example, the variables $\boldsymbol{X}_2$ and $\boldsymbol{X}_4$ can never be conditionally independent to each other given any other variables since the vertices 2 and 4 are directly connected to each other in $\mathcal{G}$. On the other hand the variable sets $\boldsymbol{X}_{2,4}$ and $\boldsymbol{X}_5$ *are* conditionally independent to each other given $\boldsymbol{Z}$ resulting in the following decomposition:

$$\mathbb{D}_{2,4,5|1,3} = \mathbb{D}_{2,4|1,3}\mathbb{D}_{5|1,3} = \mathbb{D}_{2,4|1,3}\mathbb{D}_{5|3}$$

where the condition of $\mathbb{D}_{5|1,3}$ can be reduced to $\mathbb{D}_{5|3}$ due to the variable $\boldsymbol{X}_3$ being a Markov blanket of the target variable $\boldsymbol{X}_5$. This reduces the complexity of representing the conditional distribution $\mathbb{D}_{2,4,5|1,3}$ from $\mathcal{O}(2^5)$ to $\mathcal{O}(2^4)$.

Therefore, decomposing the conditional distribution of a DM will involve finding a partition of the variables Y such that the variables in each partition set are conditionally independent, given the variables $Z = X - Y$, to the variables in any other partition set. Furthermore, finding some Markov blanket, that is a strict subset of Z, for the variables in each partition set will allow us to express and store this decomposition in a more compact manner. We will discuss how we can reuse methods from Chapter 4 for this task in Section 5.1

We then show, in Section 5.2, how this decomposition of a conditional distribution encoded in a DM assists in decomposing the conditional $\alpha\beta$-divergence between two DMs. We discuss the complexity of computing the conditional $\alpha\beta$-divergence using this decomposition while going over some examples of in Section 5.3. Lastly, we wrap up the chapter in Section 5.4.

5.1 Decomposing Conditional Distributions of Directed Graph using $\mathcal{N}$-Partitions

Recall from Example 6 that decomposing the conditional distribution of a DM $\mathbb{D}_{\mathcal{G}}$ essentially involves finding a partition of the target variables Y, and a Markov blanket for each variable set in the partition. This act of finding partitions of Y might sound similar to the concept of $\mathcal{N}$-partitions from Definition 18, where we defined them as partitions of both Y and the maximal cliques of the chordal graph $\boldsymbol{C}(\mathcal{G})$. This similarity is not a coincidence as we can show that any set N in an $\mathcal{N}$-partition $\mathcal{N}_{\mathcal{G},Y}$ is a Markov blanket of the variables in Y that are also associated with vertices in N.

Proposition 7 (Markov blanket property of $\mathcal{N}$-partitions) For any vertex set N in $\mathcal{N}$-partition $\mathcal{N}_{\mathcal{G},Y}$, through a slight abuse of notation, let Y_N be the variables in Y that are also in the partition N:

$$Y_N = Y \cap X_N$$

then, there exists some $\mathcal{N}$-partition $\mathcal{N}_{\mathcal{G},Y}$ such that for all $N \in \mathcal{N}_{\mathcal{G},Y}$, the variables X_N are a Markov blanket of the variables Y_N.

Proof We shall provide a two-step proof. The first step involves proving that the variables X_N separate the variables Y_N from all other variables $X \setminus XN$. The second step involves proving that, if Y_N is a non-empty set, then all vertices in the partition set, N, are connected to at least one vertex associated with a variable in Y that is also in N, Y_N. These two properties of N then imply that X_N is a Markov blanket of Y_N.

In order to prove that the variables $\boldsymbol{X}_N$ separate the variables $\boldsymbol{Y}_N$ from all other variables $\boldsymbol{X} \setminus \boldsymbol{X}_N$, we will provide a proof by contradiction. First assume that there exist an $N \in \boldsymbol{\mathcal{N}}_{\mathcal{G},Y}$ and $\boldsymbol{Y}_N \subseteq \boldsymbol{Y}$ such that $\boldsymbol{X}_N$ do *not* separate $\boldsymbol{Y}_N$ from variables outside $\boldsymbol{X}_N$, $\boldsymbol{X} \setminus \boldsymbol{X}_N$. This would imply that there exists a $C \in \boldsymbol{C}(\mathcal{G})$ that is not in N such that some vertex in C has an edge with the respective vertex of at least one of the variables in $\boldsymbol{Y}_N$.

$$\exists C \in \boldsymbol{C}(\mathcal{G}) \setminus \boldsymbol{C}(\mathcal{G}(N)), \exists v \in C, \exists W \in \boldsymbol{Y}_N \ : \ (v, V(W)) \in E(\mathcal{G})$$

Since C is a clique, this implies that there is a variable in $\boldsymbol{Y}_N$ whose vertex is also in C. Furthermore, from Definition 18, we know that the partitions in $\boldsymbol{\mathcal{N}}_{\mathcal{G},Y}$ are partitions of the maximal cliques $\boldsymbol{C}(\mathcal{G})$. Therefore, since C is not a subset of N, we know C is a subset of some other partition N'. This implies that N' and N shares a vertex whose variable is in $\boldsymbol{Y}_N$:

$$\exists N' \in \boldsymbol{\mathcal{N}} \ : \ \left(N' \neq N\right) \wedge \left(\boldsymbol{Y} \cap \boldsymbol{X}_N \cap \boldsymbol{X}_{N'} \neq \emptyset\right)$$

However, this *contradicts* with the definition of $\boldsymbol{\mathcal{N}}$-partitions in Definition 18, specifically, with the requirement that $\boldsymbol{\mathcal{N}}$-partitions are partitions of the vertices associated with the variables in $\boldsymbol{Y}$. Therefore, the set of variables $\boldsymbol{Y}_N \subseteq \boldsymbol{Y}$ and vertex set $N \in \boldsymbol{\mathcal{N}}$ such that N does *not* separate $\boldsymbol{Y}_N$ from $\boldsymbol{X} \setminus \boldsymbol{X}_N$ cannot exist with the partition defined in Definition 18. Conversely, the variables $\boldsymbol{X}_N$ separate the variables in $\boldsymbol{Y}_N$ from the variables in $\boldsymbol{X} \setminus \boldsymbol{X}_N$.

Now that the first step in this proof is done, the next step is to prove that all vertices in a partition set, N, are connected to at least one vertex associated with a variable in $\boldsymbol{Y}_N$, as long as $\boldsymbol{Y}_N$ is not an empty set. First assume that there is a vertex $v \in N$ that is not connected to a vertex in $V(\boldsymbol{Y}_N)$. Since vertex v must belong in some maximal clique $C \in \boldsymbol{C}(\mathcal{G}(N))$, it must be true that the clique C does not contain any vertices in $V(\boldsymbol{Y}_N)$. Therefore, we can remove the vertices in C from N, and create a new set in $\boldsymbol{\mathcal{N}}_{\mathcal{G},Y}$ containing just the vertices in C. Note that, although there might be some vertex in C that *is* connected to some vertex in $V(\boldsymbol{Y}_N)$, removing these vertices from N does not violate the definition of a $\boldsymbol{\mathcal{N}}$-partition from Definition 18. Therefore, by doing this iteratively, we can always modify a $\boldsymbol{\mathcal{N}}$-partition such that for each partition set N where $\boldsymbol{Y}_N$ is not an empty set, all vertices in N are connected to at least one vertex in $V(\boldsymbol{Y}_N)$.

Therefore, with these two facts proven, we have shown that there always exists an $\boldsymbol{\mathcal{N}}$-partition, $\boldsymbol{\mathcal{N}}_{\mathcal{G}}$, such that for any partition set $N \in \boldsymbol{\mathcal{N}}_{\mathcal{G}}$, $\boldsymbol{Y}_N \neq \emptyset$ implies that $\boldsymbol{X}_N$ is a Markov blanket of $\boldsymbol{Y}_N$.

Therefore, we can decompose the conditional distribution of a DM, $\mathbb{D}(Y \mid Z)$, as a product of conditional distributions over the sets in some $\mathcal{N}$-partition $\mathcal{N}_{\mathcal{G},Y}$:

$$\mathbb{D}(Y \mid Z) = \mathbb{D}\left(\bigcup_{N \in \mathcal{N}_{\mathcal{G},Y}} Y_N \,\middle|\, Z \right) = \prod_{N \in \mathcal{N}_{\mathcal{G},Y}} \mathbb{D}\left(Y_N \mid Z\right) = \prod_{N \in \mathcal{N}_{\mathcal{G},Y}} \mathbb{D}(Y_N \mid Z_N) \tag{5.1}$$

where, through a slight abuse of notation

$$Y_N = \emptyset \Rightarrow \forall z_N \in \mathcal{Z}_N \,:\, \mathbb{D}(Y_N \mid z_N) = 1 \tag{5.2}$$

Observe that, unlike the decomposition of marginal distributions using $\mathcal{N}$-partitions, when decomposing conditional distributions, we are only interested in the sets of $\mathcal{N}_{\mathcal{G},Y}$ that contain vertices associated with some variable in Y. Therefore, we shall define a new function, $\mathcal{B}$, where, given an $\mathcal{N}$-partition $\mathcal{N}_{\mathcal{G},Y}$, it will filter out the sets in $\mathcal{N}_{\mathcal{G},Y}$ that do not contain vertices associated with variables in Y.

Definition 23 ($\mathcal{B}$-partitions) For a chordal graph $\mathcal{G}$ and variable set $Y \subset X$, let $\mathcal{N}_{\mathcal{G},Y}$ be a $\mathcal{N}$-partition of the vertices in $\mathcal{G}$. Then $\mathcal{B}(\mathcal{N}_{\mathcal{G},Y})$ returns the set of vertex sets in $\mathcal{N}_{\mathcal{G},Y}$ that contain vertices associated with variables in Y:

$$\mathcal{B}(\mathcal{N}_{\mathcal{G},Y}) = \left\{ N \,\middle|\, N \in \mathcal{N}_{\mathcal{G},Y}, Y_N \neq \emptyset \right\} \tag{5.3}$$

Using this definition of $\mathcal{B}$-partitions, we can then express the decomposition of $\mathbb{D}(Y \mid z)$ from Equation (5.1) using only sets that are Markov blankets of the variables in Y.

Proposition 8 (Decomposition of the conditional distribution in a DM) Consider the decomposable model $\mathbb{D}_{\mathcal{G}}$ with some $\mathcal{N}$-partition $\mathcal{N}_{\mathcal{G},Y}$, where $Y \subset X$ and $Z = X - Y$. Then we can decompose the conditional distribution $\mathbb{D}(Y \mid Z)$ as:

$$\mathbb{D}(Y \mid Z) = \prod_{B \in \mathcal{B}(\mathcal{N}_{\mathcal{G},Y})} \mathbb{D}(Y_B \mid Z_B) \tag{5.4}$$

or in other words, as a product of conditional distributions over the sets in the $\mathcal{B}$-partition $\mathcal{B}(\mathcal{N}_{\mathcal{G},Y})$.

Proof

$$\mathbb{D}(Y \mid Z) = \mathbb{D}\left(\bigcup_{B \in \mathcal{B}(\mathcal{N}_{\mathcal{G},Y})} Y_B \,\middle|\, Z \right) = \prod_{B \in \mathcal{B}(\mathcal{N}_{\mathcal{G},Y})} \mathbb{D}\left(Y_B \mid Z\right) = \prod_{B \in \mathcal{B}(\mathcal{N}_{\mathcal{G},Y})} \mathbb{D}(Y_B \mid Z_B)$$

Therefore, in order to decompose $\mathbb{D}(Y \mid Z)$, we need to obtain the set of conditional distributions:

$$\left\{ \mathbb{D}(Y_B \mid Z_B) \,\middle|\, B \in \mathcal{B}(\mathcal{N}_{\mathcal{G},Y}) \right\}$$

where we can obtain each conditional distribution $\mathbb{D}(Y_B \mid Z_B)$ in this set by dividing the joint distribution over B with the marginal distribution over Z_B:

$$\mathbb{D}(Y_B \mid Z_B) = \frac{\mathbb{D}(Y_B, Z_B)}{\mathbb{D}(Z_B)} = \frac{\mathbb{D}(Y_B, Z_B)}{\sum_{Y_B \in \mathcal{Y}_B} \mathbb{D}(Y_B, Z_B)}$$

The conditional distributions $\mathbb{D}(Y_B \mid Z_B)$, for all $B \in \mathcal{B}(\mathcal{N}_{\mathcal{G},Y})$, are essentially factors over the variables X_B that can be obtained and stored with the complexity exponential in the number of vertices in B, $\mathcal{O}(2^{|B|})$.

5.2 Computing Conditional $\alpha\beta$-Divergence

Now that we have a decomposition of conditional distributions encoded in DMs, we can use this decomposition to assist in decomposing the conditional $\alpha\beta$-divergence between DMs, $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$ and $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$. Specifically in Section 5.2.1, we will first show how we can decompose the conditional functional $\mathcal{F}$ between the conditional distributions $\mathbb{P}(Y \mid Z)$ and $\mathbb{Q}(Y \mid Z)$, i.e. the expectation with respect to the marginal probability $\mathbb{P}(Z)$ of $\mathcal{F}$ between these conditional distributions. We show that the decomposition of the conditional $\mathcal{F}$ results in a sum-product over factors defined on the maximal cliques of 3 chordal graphs: $\Gamma(Z, \mathcal{N}_{\mathbb{P},Y})$, $\Gamma(X, \mathcal{B}(\mathcal{N}_{\mathbb{P},Y}))$, and $\Gamma(X, \mathcal{B}(\mathcal{N}_{\mathbb{Q},Y}))$. We also show how this sum-product is equivalent to Problem 2 and therefore can be computed using the methods in Chapter 3. Lastly, in Section 5.2.2, we will tackle computing the conditional $\alpha\beta$-divergence when the parameters $\alpha, \beta = 0$, which involves a similar procedure to computing the conditional $\mathcal{F}$, with the main difference being the factors we wish to take the sum-product of, and therefore the chordal graphs that these factors are defined over as well.

First recall from Definition 12 that we define the conditional $\alpha\beta$-divergence between models $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$ and $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$ as the expectation over the $\alpha\beta$-divergence between distributions conditioned on the values $z \in \mathcal{Z}$, with respect to the marginal distribution over the conditional variables Z.

$$\begin{aligned} D_{AB}^{(\alpha,\beta)}(\mathbb{P}_{Y|Z}, \mathbb{Q}_{Y|Z}) &= \mathbb{E}_{Z\sim\mathbb{P}}\left[D_{AB}^{(\alpha,\beta)}(\mathbb{P}_{Y|Z}, \mathbb{Q}_{Y|Z})\right] \\ &= \sum_{z\in\mathcal{Z}} \mathbb{P}_Z(z) \cdot D_{AB}^{(\alpha,\beta)}(\mathbb{P}_{Y|Z=z}, \mathbb{Q}_{Y|Z=z}) \end{aligned}$$

when both $\alpha, \beta = 0$, we can express the conditional $\alpha\beta$-divergence as such:

$$D_{AB}^{(0,0)}(\mathbb{P}_{Y|Z}, \mathbb{Q}_{Y|Z}) = \sum_{z\in\mathcal{Z}} \mathbb{P}_Z(z) \sum_{y\in\mathcal{Y}} \frac{1}{2}\left(\log \mathbb{P}_{Y|z}(y) - \log \mathbb{Q}_{Y|z}(y)\right)^2$$

On the other hand, as we will show in Proposition 9, for values other than $\alpha, \beta = 0$, we can express the conditional $\alpha\beta$-divergence as a linear combination of the

following functional:

$$\mathcal{F}_{Y|Z}(\mathbb{P},\mathbb{Q}) = \sum_{z\in\mathcal{Z}} \mathbb{P}_Z(z) \sum_{y\in\mathcal{Y}} \mathcal{F}(\mathbb{P}_{Y|z},\mathbb{Q}_{Y|z})$$

which we will call the *conditional* $\mathcal{F}_{Y|Z}$.

Proposition 9 (Conditional functional $\mathcal{F}_{Y|Z}$) Suppose we have 2 conditional distributions $\mathbb{P}_{Y|Z}$ and $\mathbb{Q}_{Y|Z}$. Then the conditional $\alpha\beta$-divergence between them

$$D_{AB}^{(\alpha,\beta)}(\mathbb{P}_{Y|Z},\mathbb{Q}_{Y|Z}) = \sum_{z\in\mathcal{Z}} \mathbb{P}_Z(z)\cdot D_{AB}^{(\alpha,\beta)}(\mathbb{P}_{Y|Z=z},\mathbb{Q}_{Y|Z=z})$$

can be expressed as a linear combination of the conditional functional $\mathcal{F}_{Y|Z}$

$$\mathcal{F}_{Y|Z}(\mathbb{P},\mathbb{Q}) = \sum_{z\in\mathcal{Z}} \mathbb{P}_Z(z) \sum_{y\in\mathcal{Y}} \mathcal{F}(\mathbb{P}_{Y|z},\mathbb{Q}_{Y|z})$$

when the parameters of the $\alpha\beta$-divergence take any values other than $\alpha,\beta = 0$.

Proof We know from Theorem 7 that, for parameters other than $\alpha,\beta = 0$, the $\alpha\beta$-divergence can be expressed as a linear combination of the functional $\mathcal{F}$

$$D_{AB}^{(\alpha,\beta)}(\mathbb{P},\mathbb{Q}) = \sum_{\mathcal{F}\in\boldsymbol{\mathcal{F}}} \mathcal{F}(\mathbb{P},\mathbb{Q})$$

where $\boldsymbol{\mathcal{F}}$ is the set of $\mathcal{F}$ parameterisations needed to express the $\alpha\beta$-divergence. We can then directly apply this fact to the conditional $\alpha\beta$-divergence:

$$\begin{aligned}
D_{AB}^{(\alpha,\beta)}(\mathbb{P}_{Y|Z},\mathbb{Q}_{Y|Z}) &= \sum_{z\in\mathcal{Z}} \mathbb{P}_Z(z)\cdot D_{AB}^{(\alpha,\beta)}(\mathbb{P}_{Y|z},\mathbb{Q}_{Y|z}) \\
&= \sum_{z\in\mathcal{Z}} \mathbb{P}_Z(z)\cdot \sum_{\mathcal{F}\in\boldsymbol{\mathcal{F}}} \mathcal{F}(\mathbb{P}_{Y|z},\mathbb{Q}_{Y|z}) \\
&= \sum_{\mathcal{F}\in\boldsymbol{\mathcal{F}}} \sum_{z\in\mathcal{Z}} \mathbb{P}_Z(z)\cdot \mathcal{F}(\mathbb{P}_{Y|z},\mathbb{Q}_{Y|z}) \\
&= \sum_{\mathcal{F}\in\boldsymbol{\mathcal{F}}} \mathcal{F}_{Y|Z}(\mathbb{P},\mathbb{Q})
\end{aligned}$$

where

$$\mathcal{F}_{Y|Z}(\mathbb{P},\mathbb{Q}) = \sum_{z\in\mathcal{Z}} \mathbb{P}_Z(z) \sum_{y\in\mathcal{Y}} \mathcal{F}(\mathbb{P}_{Y|z},\mathbb{Q}_{Y|z})$$

Therefore, the conditional $\alpha\beta$ divergence can be expressed as a linear combination of the conditional functional $\mathcal{F}_{Y|Z}$ when the parameters take values other than $\alpha,\beta = 0$.

5.2.1 Computing the Conditional Functional $\mathcal{F}_{Y|Z}$

We can start the computation of the conditional functional $\mathcal{F}_{Y,Z}$ between the DMs $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$ and $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$ by substituting their distributions into the conditional functional.

$$
\begin{aligned}
&\mathcal{F}_{Y|Z}(\mathbb{P},\mathbb{Q}) \\
&= \sum_{z\in\mathcal{Z}} \mathbb{P}_Z(z) \sum_{y\in\mathcal{Y}} g\left[\mathbb{P}_{Y|Z}\right](y,z)\cdot h\left[\mathbb{Q}_{Y|Z}\right](y,z)\cdot L\left(g^+\left[\mathbb{P}_{Y|Z}\right](y,z)\cdot h^+\left[\mathbb{Q}_{Y|Z}\right](y,z)\right) \\
&= \sum_{z\in\mathcal{Z}} \mathbb{P}_Z(z) \sum_{y\in\mathcal{Y}} g\left[\mathbb{P}_{Y|Z}\right](y,z)\cdot h\left[\mathbb{Q}_{Y|Z}\right](y,z) \\
&\qquad L\left(\left[\prod_{B\in\boldsymbol{\mathcal{B}}(\mathcal{N}_{\mathbb{P},Y})} g^+\left[\mathbb{P}_{Y_B|Z_B}\right](y,z)\right]\left[\prod_{B\in\boldsymbol{\mathcal{B}}(\mathcal{N}_{\mathbb{Q},Y})} h^+\left[\mathbb{Q}_{Y_B|Z_B}\right](y,z)\right]\right) \\
&= \sum_{B\in\boldsymbol{\mathcal{B}}(\mathcal{N}_{\mathbb{P},Y})}\sum_{x\in\mathcal{X}} L\left(g^+\left[\mathbb{P}_{Y_B|Z_B}\right](x)\right)\cdot \mathbb{P}_Z(x)\cdot g\left[\mathbb{P}_{Y|Z}\right](x)\cdot h\left[\mathbb{Q}_{Y|Z}\right](x) + \\
&\quad \sum_{B\in\boldsymbol{\mathcal{B}}(\mathcal{N}_{\mathbb{Q},Y})}\sum_{x\in\mathcal{X}} L\left(h^+\left[\mathbb{Q}_{Y_B|Z_B}\right](x)\right)\cdot \mathbb{P}_Z(x)\cdot g\left[\mathbb{P}_{Y|Z}\right](x)\cdot h\left[\mathbb{Q}_{Y|Z}\right](x) \\
&= \sum_{B\in\boldsymbol{\mathcal{B}}(\mathcal{N}_{\mathbb{P},Y})}\sum_{x\in\mathcal{X}} L\left(g^+\left[\mathbb{P}_{Y_B|Z_B}\right](x_B)\right)\cdot SP_B(x_B) + \\
&\quad \sum_{B\in\boldsymbol{\mathcal{B}}(\mathcal{N}_{\mathbb{Q},Y})}\sum_{x\in\mathcal{X}} L\left(h^+\left[\mathbb{Q}_{Y_B|Z_B}\right](x_B)\right)\cdot SP_B(x_B)
\end{aligned}
$$

where

$$
\begin{aligned}
SP_B(x_B) = \sum_{x\in\mathcal{X}_{X-B}} &\left[\prod_{N\in\boldsymbol{\mathcal{N}}_{\mathbb{P},Y}} \varphi_{N-Y}[\mathbb{P}_X](x,x_B)\right] \\
&\left[\prod_{\substack{N\in \\ \boldsymbol{\mathcal{B}}(\mathcal{N}_{\mathbb{P},Y})}} g\left[\mathbb{P}_{Y_N|Z_N}\right](x,x_B)\right]\left[\prod_{\substack{N\in \\ \boldsymbol{\mathcal{B}}(\mathcal{N}_{\mathbb{Q},Y})}} h\left[\mathbb{Q}_{Y_N|Z_N}\right](x,x_B)\right]
\end{aligned} \tag{5.5}
$$

Similar to computing the ordinary functional $\mathcal{F}$ between joint and marginal distributions of DMs, computing the conditional functional $\mathcal{F}_{Y|Z}$ involves obtaining the result of a set of sum-products over factors defined over the maximal cliques of multiple chordal graphs.

The difference between computing the conditional functional $\mathcal{F}_{Y|Z}$ and the ordinary functional $\mathcal{F}$ between DMs is that the sum-products we need for computing $\mathcal{F}_{Y|Z}$ is a sum over factors defined over three different chordal graphs, and not just two.

Specifically, the factors that SP_B sums over are

$$\begin{aligned}
\Phi_Z &:= \left\{ \varphi_N[\mathbb{P}_X] \;\middle|\; N \in \boldsymbol{C}\left(\Gamma\left(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathbb{P},Y}\right)\right) \right\} \\
\Phi_{\mathbb{P}} &:= \left\{ g\left[\mathbb{P}_{Y_N|Z_N}\right] \;\middle|\; N \in \boldsymbol{C}\left(\Gamma\left(\boldsymbol{X}, \boldsymbol{\mathcal{B}}\left(\boldsymbol{\mathcal{N}}_{\mathbb{P},Y}\right)\right)\right) \right\} \\
\Phi_{\mathbb{Q}} &:= \left\{ h\left[\mathbb{Q}_{Y_N|Z_N}\right] \;\middle|\; N \in \boldsymbol{C}\left(\Gamma\left(\boldsymbol{X}, \boldsymbol{\mathcal{B}}\left(\boldsymbol{\mathcal{N}}_{\mathbb{Q},Y}\right)\right)\right) \right\}
\end{aligned} \tag{5.6}$$

Recall that $\Gamma\left(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathbb{P},Y}\right)$ is just the chordal graph, containing only vertices for variables in $\boldsymbol{Z}$, that we used for computing the marginal $\alpha\beta$-divergence in Chapter 4. Furthermore, recall that the function Γ from Definition 20 returns a chordal graph with vertices that are associated with some variable in the first argument, and also in the $\boldsymbol{\mathcal{N}}$-partition or $\boldsymbol{\mathcal{B}}$-partition given in the second argument. Therefore, the chordal graphs $\Gamma\left(\boldsymbol{X}, \boldsymbol{\mathcal{B}}\left(\boldsymbol{\mathcal{N}}_{\mathbb{P},Y}\right)\right)$ and $\Gamma\left(\boldsymbol{X}, \boldsymbol{\mathcal{B}}\left(\boldsymbol{\mathcal{N}}_{\mathbb{P},Y}\right)\right)$ only contain vertices connected to at least one vertex associated with the variables in $\boldsymbol{Y}$, and not all the vertices in the graph structure of the original DMs $\mathbb{P}_{\mathcal{G}_{\mathbb{P}}}$ and $\mathbb{Q}_{\mathcal{G}_{\mathbb{Q}}}$.

More importantly, observe that the factors in Φ_Z and $\Phi_{\mathbb{P}}$ are defined over chordal graphs that based on the graph structure of DM $\mathbb{P}_{\mathcal{G}_{\mathbb{P}}}$. Therefore, we can merge these sets of factors, resulting in a new set of factors that are defined over the maximal cliques of a new chordal graph.

Theorem 12 (Merging factors that share the same supergraph)
Suppose we have the DM $\mathbb{D}_{\mathcal{G}}$ with target variables $\boldsymbol{Y}$ and conditional variables $\boldsymbol{Z}$ such that $\boldsymbol{Y} \cup \boldsymbol{Z} = \boldsymbol{X}$. Furthermore, assume we have an $\boldsymbol{\mathcal{N}}$-partition $\boldsymbol{\mathcal{N}}_{\mathcal{G},Y}$, and the following sets of factors defined over the maximal cliques of $\boldsymbol{\mathcal{N}}$-graphs based on $\boldsymbol{\mathcal{N}}_{\mathcal{G},Y}$.

$$\begin{aligned}
\Phi_1 &:= \left\{ \phi_{1,C} \;\middle|\; C \in \boldsymbol{C}\left(\Gamma\left(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y}\right)\right) \right\} \\
\Phi_2 &:= \left\{ \phi_{2,C} \;\middle|\; C \in \boldsymbol{C}\left(\Gamma\left(\boldsymbol{X}, \boldsymbol{N}\right)\right) \right\}
\end{aligned}$$

where

$$\boldsymbol{N} \subseteq \boldsymbol{\mathcal{N}}_{\mathcal{G},Y}$$

Then there exists a new set of factors Φ defined on a new chordal graph $\mathcal{G}'$

$$\Psi := \left\{ \psi_C \;\middle|\; C \in \boldsymbol{C}\left(\mathcal{G}'\right) \right\}$$

such that

$$\prod_{\psi \in \Psi} \psi = \prod_{\phi_1 \in \Phi_1} \phi_1 \prod_{\phi_2 \in \Phi_2} \phi_2$$

Proof First of all, observe that a subset of factors in Φ_1 is also defined over variables in the sets of $\boldsymbol{N} \subseteq \boldsymbol{\mathcal{N}}_{\mathcal{G},Y}$. However, unlike Φ_2, factors in Φ_1 are only defined over the variables in $\boldsymbol{Z}$, and not all the variables in $\boldsymbol{X}$. Therefore, all factors in Φ_2 will have a strictly larger domain than exactly one factor in Φ_1. As a result, we can merge the factors in Φ_2 with a subset of the factors in Φ_1 by multiplying the corresponding factors.

$$\left\{ \phi_{2,C} \cdot \phi_{1,C-Y} \;\middle|\; C \in \boldsymbol{C}\left(\Gamma\left(\boldsymbol{X}, \boldsymbol{N}\right)\right) \right\}$$

This results in the remaining set of factors from Φ_1 that we can add separately to Ψ:

$$\left\{ \phi_{1,C} \;\middle|\; C \in \boldsymbol{C}\left(\Gamma\left(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y} \setminus \boldsymbol{N}\right)\right) \right\}$$

Therefore, the final set of factors, Ψ, obtained from merging Φ_1 and Φ_2 is

$$\Psi := \left\{ \phi_{2,C} \cdot \phi_{1,C-Y} \;\middle|\; C \in \boldsymbol{C}\left(\Gamma\left(\boldsymbol{X}, \boldsymbol{N}\right)\right) \right\} \bigcup \left\{ \phi_{1,C} \;\middle|\; C \in \boldsymbol{C}\left(\Gamma\left(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y} \setminus \boldsymbol{N}\right)\right) \right\}$$

We can then construct the chordal graph $\mathcal{G}'$ by using a method similar to the method used to construct Ψ. That is to say, we construct $\mathcal{G}'$ by taking the graph union of $\Gamma(\boldsymbol{X}, \boldsymbol{N})$ and $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y} \setminus \boldsymbol{N})$.

$$\mathcal{G}' := \Gamma(\boldsymbol{X}, \boldsymbol{N}) \bigcup \Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y} \setminus \boldsymbol{N})$$

We know $\mathcal{G}'$ is chordal as it is equivalent to taking the chordal graph $\Gamma(\boldsymbol{X}, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y})$, and removing any vertices that is associated with a variable in $\boldsymbol{Y}$ and not in $\boldsymbol{N}$, and, as we know from Corollary 1, vertex deletion maintains chordality. Furthermore, since by definition, any vertex associated with a variable in $\boldsymbol{Y}$ is not in any minimal separator of $\Gamma(\boldsymbol{X}, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y})$, the deletion of these vertices only removes these vertices from the set of existing maximal cliques, leaving these maximal cliques otherwise unchanged. Therefore, the maximal cliques of $\mathcal{G}'$ are exactly just the union of the maximal cliques of $\Gamma(\boldsymbol{X}, \boldsymbol{N})$ and $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y} \setminus \boldsymbol{N})$.

As a result, since the factors of Ψ are defined over the maximal cliques of $\Gamma(\boldsymbol{X}, \boldsymbol{N})$ and $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y} \setminus \boldsymbol{N})$, these factors are also defined over the maximal

cliques of $\mathcal{G}'$.

$$\begin{aligned}\Psi &:= \left\{ \mathcal{I}_{\boldsymbol{N}}(C) \cdot \phi_{2,C} \cdot \phi_{1,C-Y} + \big(1 - \mathcal{I}_{\boldsymbol{N}}(C)\big) \cdot \phi_{1,C-Y} \;\middle|\; C \in \boldsymbol{C}(\mathcal{G}') \right\} \\ &= \left\{ \Big(\mathcal{I}_{\boldsymbol{N}}(C) \cdot \phi_{2,C} + \big(1 - \mathcal{I}_{\boldsymbol{N}}(C)\big) \Big) \cdot \phi_{1,C-Y} \;\middle|\; C \in \boldsymbol{C}(\mathcal{G}') \right\}\end{aligned}$$

where $\mathcal{I}_{\boldsymbol{N}}$ is the indicator function

$$\mathcal{I}_{\boldsymbol{N}}(C) = \begin{cases} 1 & C \in \boldsymbol{N} \\ 0 & \text{otherwise} \end{cases}$$

Thus, we can use Theorem 12 to merge the factor sets Φ_Z and $\Phi_{\mathbb{P}}$ from Equation (5.6) into a single factor set, $\Psi_{\mathbb{P}}$, containing factors defined over the maximal cliques of the graph union of $\Gamma(\boldsymbol{X}, \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathcal{G},Y}))$ and $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathcal{G},Y} \setminus \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathcal{G},Y}))$. Therefore, we now have 2 sets of factors:

$$\Psi_{\mathbb{P}} := \left\{ \Big(\mathcal{I}_{\boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathcal{G},Y})}(C) \cdot g\big[\mathbb{P}_{Y_N|Z_N}\big] + \big(1 - \mathcal{I}_{\boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathcal{G},Y})}(C)\big) \Big) \cdot \varphi_{C-Y}\big[\mathbb{P}_X\big] \;\middle|\; C \in \boldsymbol{C}(\mathcal{G}'_{\mathbb{P}}) \right\}$$

$$\Psi_{\mathbb{Q}} := \Phi_{\mathbb{Q}} = \left\{ h\big[\mathbb{Q}_{Y_C|Z_C}\big] \;\middle|\; C \in \boldsymbol{C}(\mathcal{G}'_{\mathbb{Q}}) \right\}$$

where

$$\begin{aligned}\mathcal{G}'_{\mathbb{P}} &:= \Gamma(\boldsymbol{X}, \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathcal{G},Y})) \bigcup \Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathbb{P},Y} \setminus \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathcal{G},Y})) \\ \mathcal{G}'_{\mathbb{Q}} &:= \Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathbb{Q},Y})\end{aligned}$$

Observe that the factors in $\Psi_{\mathbb{P}}$ and $\Psi_{\mathbb{Q}}$ are defined over the maximal cliques of the chordal graphs $\mathcal{G}'_{\mathbb{P}}$ and $\mathcal{G}'_{\mathbb{Q}}$ respectively. Using these set of factors, we can now compute the sum-product $SP_B(\boldsymbol{x}_B)$ from Equation (5.5)

$$\begin{aligned}&SP_B(\boldsymbol{x}_B) \\ &= \sum_{\boldsymbol{x} \in \mathcal{X}_{X-B}} \left[\prod_{N \in \boldsymbol{\mathcal{N}}_{\mathbb{P},Y}} \varphi_{N-Y}[\mathbb{P}_X](\boldsymbol{x}, \boldsymbol{x}_B) \right] \left[\prod_{\substack{N \in \\ \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathbb{P},Y})}} g\big[\mathbb{P}_{Y_N|Z_N}\big](\boldsymbol{x}, \boldsymbol{x}_B) \right] \left[\prod_{\substack{N \in \\ \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathbb{Q},Y})}} h\big[\mathbb{Q}_{Y_N|Z_N}\big](\boldsymbol{x}, \boldsymbol{x}_B) \right] \\ &= \sum_{\boldsymbol{x} \in \mathcal{X}_{X-B}} \left[\prod_{C \in \boldsymbol{C}(\mathcal{G}'_{\mathbb{P}})} \psi_{\mathbb{P},C}(\boldsymbol{x}, \boldsymbol{x}_B) \right] \left[\prod_{C \in \boldsymbol{C}(\mathcal{G}'_{\mathbb{Q}})} \psi_{\mathbb{Q},C}(\boldsymbol{x}, \boldsymbol{x}_B) \right]\end{aligned}$$

with MGASP from Section 3.2. As a result, the complexity of obtaining all the SP_B we need to compute the conditional functional $\mathcal{F}_{Y|Z}$ is the same complexity as using MGASP with chordal graphs $\mathcal{G}'_{\mathbb{P}}$ and $\mathcal{G}'_{qr}$, which as we have showed in Section 3.4, is

$$\mathcal{O}\big(|\boldsymbol{X}| \cdot 2^{\omega(\mathcal{H}')+1}\big)$$

where $\mathcal{H}'$ is the computation graph of $\mathcal{G}'_{\mathbb{P}}$ and $\mathcal{G}'_{qr}$.

5.2.2 Computing the Conditional $\alpha\beta$-Divergence when $\alpha, \beta = 0$

So far we have shown how to use MGASP from Section 3.2 to compute the conditional functional $\mathcal{F}$, and therefore the conditional $\alpha\beta$-divergence for parameters other than $\alpha, \beta = 0$, between two DMs. In this section, we will show how to do the same for the case when $\alpha, \beta = 0$. First recall the conditional $\alpha\beta$-divergence when $\alpha, \beta = 0$:

$$
\begin{aligned}
&D_{AB}^{(0,0)}(\mathbb{P}_{Y|Z}, \mathbb{Q}_{Y|Z}) \\
&= \sum_{z \in \mathcal{Z}} \mathbb{P}_Z(z) \sum_{y \in \mathcal{Y}} \frac{1}{2} \left(\log \mathbb{P}_{Y|z}(y) - \log \mathbb{Q}_{Y|z}(y)\right)^2 \\
&= \frac{1}{2} \sum_{z \in \mathcal{Z}} \mathbb{P}_Z(z) \sum_{y \in \mathcal{Y}} \left(\log \mathbb{P}_{Y|z}(y)\right)^2 + \left(\log \mathbb{Q}_{Y|z}(y)\right)^2 - \left(\log \mathbb{P}_{Y|z}(y) \log \mathbb{Q}_{Y|z}(y)\right)
\end{aligned}
$$

In an effort to generalise this expression, we shall define a functional $\mathcal{F}_{Y|Z}^{(0,0)}$:

$$
\mathcal{F}_{Y|Z}^{(0,0)}(\mathbb{D}^{(1)}, \mathbb{D}^{(2)}) = \frac{1}{2} \sum_{z \in \mathcal{Z}} \mathbb{P}_Z(z) \sum_{y \in \mathcal{Y}} \log \mathbb{D}_{Y|z}^{(1)}(y) \log \mathbb{D}_{Y|z}^{(2)}(y) \tag{5.7}
$$

resulting in the following form for $D_{AB}^{(0,0)}(\mathbb{P}_{Y|Z}, \mathbb{Q}_{Y|Z})$:

$$
D_{AB}^{(0,0)}(\mathbb{P}_{Y|Z}, \mathbb{Q}_{Y|Z}) = \mathcal{F}_{Y|Z}^{(0,0)}(\mathbb{P}, \mathbb{P}) + \mathcal{F}_{Y|Z}^{(0,0)}(\mathbb{Q}, \mathbb{Q}) - \mathcal{F}_{Y|Z}^{(0,0)}(\mathbb{P}, \mathbb{Q})
$$

Substituting the distributions $\mathbb{D}^{(1)}$ and $\mathbb{D}^{(2)}$ for the distributions of DMs $\mathbb{D}_{\mathcal{G}_1}^{(1)}$ and $\mathbb{D}_{\mathcal{G}_2}^{(2)}$ we get:

$$
\begin{aligned}
&\mathcal{F}_{Y|Z}^{(0,0)}(\mathbb{D}^{(1)}, \mathbb{D}^{(2)}) \\
&= \frac{1}{2} \sum_{z \in \mathcal{Z}} \mathbb{P}_Z(z) \sum_{y \in \mathcal{Y}} \log \mathbb{D}_{Y|Z}^{(1)}(y \mid z) \log \mathbb{D}_{Y|Z}^{(2)}(y \mid z) \\
&= \frac{1}{2} \sum_{z \in \mathcal{Z}} \left(\prod_{N \in \mathcal{N}_{\mathcal{G}_{\mathbb{P}}, Y}} \varphi_{N-Y}[\mathbb{P}_X](z) \right) \\
&\qquad \sum_{y \in \mathcal{Y}} \left(\sum_{N \in \mathcal{B}(\mathcal{N}_{\mathbb{D}^{(1)}, Y})} \log \mathbb{D}_{Y_N|Z_N}^{(1)}(y_N \mid z_N) \right) \left(\sum_{N \in \mathcal{B}(\mathcal{N}_{\mathbb{D}^{(2)}, Y})} \log \mathbb{D}_{Y_{N'}|Z_{N'}}^{(2)}(y_{N'} \mid z_{N'}) \right) \\
&= \frac{1}{2} \sum_{B^+ \in \mathcal{B}(\mathcal{N}_{\mathbb{D}^{(1)}, Y})} \sum_{B^* \in \mathcal{B}(\mathcal{N}_{\mathbb{D}^{(2)}, Y})}
\end{aligned}
$$

$$\sum_{x\in\mathcal{X}}\left(\prod_{N\in\boldsymbol{\mathcal{N}}_{\mathcal{G}_{\mathbb{P}},Y}}\varphi_{N-Y}[\mathbb{P}_X](\boldsymbol{x}_Z)\right)\left(\log\mathbb{D}^{(1)}_{Y_{B^+}|Z_{B^+}}(\boldsymbol{x}_{B^+})\right)\left(\log\mathbb{D}^{(2)}_{Y_{B^*}|Z_{B^*}}(\boldsymbol{x}_{B^*})\right)$$

We then have the following sum-product for all $B^+ \in \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathbb{D}^{(1)},Y})$ and $B^* \in \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathbb{D}^{(2)},Y})$:

$$\sum_{x\in\mathcal{X}}\left(\prod_{N\in\boldsymbol{\mathcal{N}}_{\mathcal{G}_{\mathbb{P}},Y}}\varphi_{N-Y}[\mathbb{P}_X](\boldsymbol{x}_Z)\right)\left(\log\mathbb{D}^{(1)}_{Y_{N^+}|Z_{N^+}}(\boldsymbol{x}_N)\right)\left(\log\mathbb{D}^{(2)}_{Y_{N^*}|Z_{N^*}}(\boldsymbol{x}_{N'})\right)$$

which can expressed as the sum-product of three sets of factors defined over the maximal cliques of their respective chordal graphs:

$$\begin{aligned}
\Phi_Z &:= \left\{\varphi_C[\mathbb{P}_X] \;\middle|\; C\in\Gamma(Z,\boldsymbol{\mathcal{N}}_{\mathbb{P},Y})\right\}\\
\Phi_1 &:= \left\{\log\mathbb{D}^{(1)}_{Y_C|Z_C} \;\middle|\; C\in\boldsymbol{C}(\Gamma(B^+,\boldsymbol{\mathcal{N}}_{\mathbb{D}^{(1)},Y}))\right\} = \left\{\log\mathbb{D}^{(1)}_{Y_{B^+}|Z_{B^+}}\right\}\\
\Phi_2 &:= \left\{\log\mathbb{D}^{(2)}_{Y_C|Z_C} \;\middle|\; C\in\boldsymbol{C}(\Gamma(B^*,\boldsymbol{\mathcal{N}}_{\mathbb{D}^{(2)},Y}))\right\} = \left\{\log\mathbb{D}^{(2)}_{Y_{B^*}|Z_{B^*}}\right\}
\end{aligned} \tag{5.8}$$

these three factor sets can be further merged into just two factor sets depending on the distributions $\mathbb{D}^{(1)}$ and $\mathbb{D}^{(2)}$ passed to $\mathcal{F}^{(0,0)}_{Y|Z}$. Therefore, for the remainder of this section, we will go through the 3 different cases: $\mathcal{F}^{(0,0)}_{Y|Z}(\mathbb{Q},\mathbb{Q})$, $\mathcal{F}^{(0,0)}_{Y|Z}(\mathbb{P},\mathbb{P})$, and $\mathcal{F}^{(0,0)}_{Y|Z}(\mathbb{P},\mathbb{Q})$.

1. In the case for $\mathcal{F}^{(0,0)}_{Y|Z}(\mathbb{Q},\mathbb{Q})$, the factor sets Φ_1 and Φ_2 can be merged directly, resulting in the final 2 merged factor sets

$$\begin{aligned}
\Psi_{\mathbb{P}} &:= \left\{\varphi_C[\mathbb{P}_X] \;\middle|\; C\in\Gamma(Z,\boldsymbol{\mathcal{N}}_{\mathbb{P},Y})\right\}\\
\Psi_{\mathbb{Q}} &:= \left\{\log\mathbb{Q}_{Y_C|Z_C} \;\middle|\; C\in\boldsymbol{C}(\Gamma(B^*\cup B^*,\boldsymbol{\mathcal{N}}_{\mathbb{Q},Y}))\right\}
\end{aligned}$$

   both defined over the maximal cliques of chordal graphs $\Gamma(Z,\boldsymbol{\mathcal{N}}_{\mathbb{P},Y})$ and $\Gamma(B^*\cup B^*,\boldsymbol{\mathcal{N}}_{\mathbb{Q},Y})$. Therefore we can compute the sum-product over factors in $\Psi_{\mathbb{P}}$ and $\Psi_{\mathbb{Q}}$ using MGASP.

2. For $\mathcal{F}^{(0,0)}_{Y|Z}(\mathbb{P},\mathbb{P})$, we can actually merge the three factor sets in Equation (5.8) into just a single factor set using Theorem 12.

$$\Psi := \left\{\left(\boldsymbol{\mathcal{I}}_{\boldsymbol{N}}(C)\cdot\log\mathbb{P}_{Y_N|Z_N} + \left(1-\boldsymbol{\mathcal{I}}_{\boldsymbol{N}}(C)\right)\right)\cdot\varphi_{C-Y}\left[\mathbb{P}_X\right] \;\middle|\; C\in\boldsymbol{C}(\mathcal{G}'_{\mathbb{P}})\right\}$$

   where

$$\begin{aligned}
\boldsymbol{N} &:= \left\{B^+,B^*\right\}\\
\mathcal{G}'_{\mathbb{P}} &:= \Gamma(X,\boldsymbol{N})\bigcup\Gamma(Z,\boldsymbol{\mathcal{N}}_{\mathbb{P},Y}\setminus\boldsymbol{N})
\end{aligned}$$

The sum over the product of factors in Ψ can then be computed using either variable elimination or the JTA over the chordal graph $\mathcal{G}'_{\mathbb{P}}$.

3. Finally, for $\mathcal{F}^{(0,0)}_{Y|Z}(\mathbb{P},\mathbb{Q})$, we can use Theorem 12 to merge factor sets Φ_Z and Φ_1 in Equation (5.8) resulting in the two sets of factors:

$$\Psi_{\mathbb{P}} := \left\{ \left(\mathcal{I}_{\boldsymbol{N}}(C) \cdot \log \mathbb{P}_{Y_N|Z_N} + \left(1 - \mathcal{I}_{\boldsymbol{N}}(C)\right) \right) \cdot \varphi_{C-Y}\left[\mathbb{P}_X\right] \;\middle|\; C \in \boldsymbol{C}(\mathcal{G}'_{\mathbb{P}}) \right\}$$
$$\Psi_{\mathbb{Q}} := \left\{ \log \mathbb{D}^{(2)}_{Y_{B^*}|Z_{B^*}} \right\}$$

where

$$\boldsymbol{N} := \left\{ B^+ \right\}$$
$$\mathcal{G}'_{\mathbb{P}} := \Gamma(\boldsymbol{X},\boldsymbol{N}) \bigcup \Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathbb{P},Y} \setminus \boldsymbol{N})$$

Since factor sets $\Psi_{\mathbb{P}}$ and $\Psi_{\mathbb{Q}}$ are defined over the maximal cliques of chordal graphs $\mathcal{G}'_{\mathbb{P}}$ and $\Gamma(B^*, \boldsymbol{\mathcal{N}}_{\mathbb{Q},Y})$, we can use MGASP to compute the sum-product over the factors of these two factor sets.

Ultimately, the conditional $\alpha\beta$-divergence between two DMs, $\mathbb{P}_{\mathcal{G}_{\mathbb{P}}}$ and $\mathbb{Q}_{\mathcal{G}_{\mathbb{Q}}}$, when $\alpha, \beta = 0$ can be computed, by computing

$$D^{(0,0)}_{AB}(\mathbb{P}_{Y|Z}, \mathbb{Q}_{Y|Z}) = \mathcal{F}^{(0,0)}_{Y|Z}(\mathbb{P},\mathbb{P}) + \mathcal{F}^{(0,0)}_{Y|Z}(\mathbb{Q},\mathbb{Q}) - \mathcal{F}^{(0,0)}_{Y|Z}(\mathbb{P},\mathbb{Q})$$

where

$$\mathcal{F}^{(0,0)}_{Y|Z}(\mathbb{D}^{(1)},\mathbb{D}^{(2)}) = \frac{1}{2} \sum_{z \in \mathcal{Z}} \mathbb{P}_Z(z) \sum_{y \in \mathcal{Y}} \log \mathbb{D}^{(1)}_{Y|z}(y) \log \mathbb{D}^{(2)}_{Y|z}(y)$$

by using a combination of the JTA with MGASP from Section 3.2. Observe, that regardless of what the distributions, $\mathbb{D}^{(1)}$ and $\mathbb{D}^{(2)}$, given to $\mathcal{F}^{(0,0)}_{Y|Z}$ are, the complexity of computing the needed sum-products is upper bounded by $\mathcal{O}\left(2^{\omega(\mathcal{H})+1}\right)$, where $\mathcal{H}$ is the computation graph between $\Gamma(\boldsymbol{X}, \boldsymbol{\mathcal{N}}_{\mathbb{P},Y})$ and $\Gamma(\boldsymbol{X}, \boldsymbol{\mathcal{N}}_{\mathbb{Q},Y})$. Therefore, the complexity of computing the conditional $\alpha\beta$-divergence between $\mathbb{P}_{\mathcal{G}_{\mathbb{P}}}$ and $\mathbb{Q}_{\mathcal{G}_{\mathbb{Q}}}$ when $\alpha, \beta = 0$ is:

$$\mathcal{O}\left(\max\left(\left|\boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathbb{D}^{(1)},Y})\right|, \left|\boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathbb{D}^{(2)},Y})\right|\right)^2 \cdot 2^{\omega(\mathcal{H})+1}\right) \in \mathcal{O}\left(|\boldsymbol{X}|^2 \cdot 2^{\omega(\mathcal{H})+1}\right)$$

since we need to compute the sum-product for each $B^+ \in \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathbb{D}^{(1)},Y})$ and $B^* \in \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathbb{D}^{(2)},Y})$.

5.3 Complexity

As usual, we can determine the computational complexity of computing the conditional $\alpha\beta$-divergence by first checking, in $\mathcal{O}(1)$ time, what the given values for α and β are. When $\alpha, \beta = 0$, we have showed in Section 5.2.2 that the complexity of computing $D_{AB}^{(0,0)}(\mathbb{P}_{Y|Z}, \mathbb{Q}_{Y|Z})$ is:

$$D_{AB}^{(0,0)}(\mathbb{P}_{Y|Z}, \mathbb{Q}_{Y|Z}) \in \mathcal{O}\left(|\boldsymbol{X}|^2 2^{\omega(\mathcal{H})+1}\right)$$

where $\mathcal{H}$ is the computation graph between $\Gamma(\boldsymbol{X}, \boldsymbol{\mathcal{N}}_{\mathbb{P},Y})$ and $\Gamma(\boldsymbol{X}, \boldsymbol{\mathcal{N}}_{\mathbb{Q},Y})$.

In cases other than $\alpha, \beta = 0$, we reframed the problem into one of computing the $\alpha\beta$-divergence between a product of factors defined over the maximal cliques of the chordal graphs $\mathcal{G}'_{\mathbb{P}}$ and $\mathcal{G}'_{\mathbb{Q}}$:

$$\Psi_{\mathbb{P}} := \left\{ \left(\mathcal{I}_{\boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathcal{G},Y})}(C) \cdot g\left[\mathbb{P}_{Y_N|Z_N}\right] + \left(1 - \mathcal{I}_{\boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathcal{G},Y})}(C)\right) \right) \cdot \varphi_{C-Y}\left[\mathbb{P}_X\right] \;\middle|\; C \in \boldsymbol{C}(\mathcal{G}'_{\mathbb{P}}) \right\}$$

$$\Psi_{\mathbb{Q}} := \Phi_{\mathbb{Q}} = \left\{ h\left[\mathbb{Q}_{Y_C|Z_C}\right] \;\middle|\; C \in \boldsymbol{C}(\mathcal{G}'_{\mathbb{Q}}) \right\}$$

where

$$\mathcal{G}'_{\mathbb{P}} := \Gamma(\boldsymbol{X}, \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathcal{G},Y})) \bigcup \Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathbb{P},Y} \setminus \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathcal{G},Y}))$$

$$\mathcal{G}'_{\mathbb{Q}} := \Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathbb{Q},Y})$$

Using this transformation of the problem, we can use MGASP from Section 3.2 to compute conditional $\alpha\beta$-divergence when the parameters take values other than $\alpha, \beta = 0$. Using MGASP with the factors defined over chordal graphs $\mathcal{G}'_{\mathbb{P}}$ and $\mathcal{G}'_{\mathbb{Q}}$ results in the computational complexity

$$\mathcal{O}\left(|\boldsymbol{X}| \cdot 2^{\omega(\mathcal{H}')+1}\right)$$

where $\mathcal{H}'$ is the computation graph of $\mathcal{G}'_{\mathbb{P}}$ and $\mathcal{G}'_{\mathbb{Q}}$.

Therefore, the complexity of computing the conditional $\alpha\beta$-divergence between the DMs $\mathbb{P}_{\mathcal{G}_{\mathbb{P}}}$ and $\mathbb{Q}_{\mathcal{G}_{\mathbb{Q}}}$ in general is

$$D_{AB}^{(\alpha,\beta)}(\mathbb{P}_{Y|Z}, \mathbb{Q}_{Y|Z}) \in \begin{cases} \mathcal{O}\left(|\boldsymbol{X}|^2 \cdot 2^{\omega(\mathcal{H})+1}\right) & \alpha, \beta = 0 \\ \mathcal{O}\left(|\boldsymbol{X}| \cdot 2^{\omega(\mathcal{H}')+1}\right) & \text{otherwise} \end{cases}$$

5.4 Conclusion

In this chapter we first showed how to decompose the conditional $\alpha\beta$-divergence between two DMs, $D_{AB}^{(\alpha,\beta)}(\mathbb{P}_{Y|Z}, \mathbb{Q}_{Y|Z})$, by finding a decomposition of conditional distributions encoded in some DM. We then showed how we can use this decomposition to efficiently compute the conditional $\alpha\beta$-divergence between two DMs.

Decomposing the conditional distribution of a DM $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$, $\mathbb{P}_{Y|Z}$, involves first finding a partition of the variables in $\boldsymbol{Y}$ such that the $\boldsymbol{Y}$ variables of each set in the partition have a Markov blanket that makes them conditionally independent to $\boldsymbol{Y}$ in other partition sets, given this Markov blanket. Such partitions of $\boldsymbol{Y}$ can be obtained by just finding some $\boldsymbol{\mathcal{N}}$-partitions, $\boldsymbol{\mathcal{N}}_{\mathcal{G}_{pr},Y}$. In fact, we showed in Proposition 7 that the variables in each set of an $\boldsymbol{\mathcal{N}}$-partition is a Markov blanket of the $\boldsymbol{Y}$ variables in the partition set. With the $\boldsymbol{\mathcal{N}}$-partition $\boldsymbol{\mathcal{N}}_{\mathcal{G}_\mathbb{P},Y}$, we can then use the conditional independence property to decompose the conditional distribution $\mathbb{P}_{Y|Z}$, into a product of smaller conditional distributions over the sets in $\boldsymbol{\mathcal{N}}_{\mathcal{G}_\mathbb{P},Y}$.

With a decomposition of the conditional distributions $\mathbb{P}_{Y|Z}$ and $\mathbb{Q}_{Y|Z}$ for DMs $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$ and $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$, we then attempted to decompose the conditional $\alpha\beta$-divergence between these conditional distributions. However, the $\alpha\beta$-divergence has different expressions depending on the values of α and β. Therefore, we tackled decomposing the conditional $\alpha\beta$-divergence in the case when either α or β is 0 and when α, β=0.

When either α or β is 0, we can express the ordinary $\alpha\beta$-divergence as a linear combination of the functional $\mathcal{F}$ from Definition 13. Therefore, we defined a conditional version of $\mathcal{F}$ that we called the conditional functional $\mathcal{F}_{Y|Z}$, and showed, in Proposition 9, that the conditional $\alpha\beta$-divergence can be expressed as a linear combination of these conditional functionals $\mathcal{F}_{Y|Z}$. We then showed that computing $\mathcal{F}_{Y|Z}$ mainly involves computing the sum-product over factors defined on the maximal cliques of three different chordal graphs: $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathbb{P},Y})$, $\Gamma(\boldsymbol{X}, \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathbb{P},Y}))$, and $\Gamma(\boldsymbol{X}, \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathbb{Q},Y}))$. However, since the chordal graphs $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathbb{P},Y})$ and $\Gamma(\boldsymbol{X}, \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathbb{P},Y}))$ are both subgraphs of $\mathcal{G}_\mathbb{P}$, we showed in Theorem 12 that we can directly merge these chordal graphs and their associated factor sets to form a single factor set $\Psi_\mathbb{P}$ defined over the maximal cliques of the chordal graph $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathbb{P},Y}) \cup \Gamma(\boldsymbol{X}, \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathbb{P},Y}))$. This merger results in 2 sets of factors, defined over the maximal cliques of 2 chordal graphs respectively, of which we wish to compute the sum-product, the same problem described in Problem 2 way back in Chapter 3. Due to this similarity, we are able to use this transformation to compute the conditional $\alpha\beta$-divergence, when either α or β is 0, between two DMs in a time complexity of $\mathcal{O}\left(|\boldsymbol{X}| \cdot 2^{\omega(\mathcal{H}')+1}\right)$, where $\mathcal{H}'$ is the computation graph of $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathbb{P},Y}) \cup \Gamma(\boldsymbol{X}, \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathbb{P},Y}))$ and $\Gamma(\boldsymbol{Z}, \boldsymbol{\mathcal{N}}_{\mathbb{P},Y})$.

Furthermore, we showed that a similar procedure can be used to compute the conditional $\alpha\beta$-divergence when both α and β are 0. Specifically, we first defined a new functional, $\mathcal{F}_{Y|Z}^{(0,0)}$, and showed how the conditional $\alpha\beta$-divergence between $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$

and $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$, when $\alpha, \beta = 0$, can be expressed as a linear combination of $\mathcal{F}^{(0,0)}_{Y|Z}(\mathbb{P}, \mathbb{P})$, $\mathcal{F}^{(0,0)}_{Y|Z}(\mathbb{Q}, \mathbb{Q})$, and $\mathcal{F}^{(0,0)}_{Y|Z}(\mathbb{P}, \mathbb{Q})$. Similar to $\mathcal{F}_{Y|Z}$, the main computation component of $\mathcal{F}^{(0,0)}_{Y|Z}$ involves the sum-product over factors defined on the maximal cliques of three chordal graphs. However, unlike $\mathcal{F}_{Y|Z}$, it is possible for $\mathcal{F}^{(0,0)}_{Y|Z}$ to be passed arguments other than $(\mathbb{P}, \mathbb{Q})$. Therefore, we had to tackle merging these three factor sets for the case of $\mathcal{F}^{(0,0)}_{Y|Z}(\mathbb{P}, \mathbb{P})$, $\mathcal{F}^{(0,0)}_{Y|Z}(\mathbb{Q}, \mathbb{Q})$, and $\mathcal{F}^{(0,0)}_{Y|Z}(\mathbb{P}, \mathbb{Q})$ separately. Ultimately, we showed that in all cases, $\mathcal{F}^{(0,0)}_{Y|Z}$ can be reframed into a form that is amendable for computation by either the ordinary JTA or by MGASP. As such, the complexity of computing the conditional $\alpha\beta$-divergence when both α and β are 0 is $\mathcal{O}\left(|\boldsymbol{X}|^2 \cdot 2^{\omega(\mathcal{H})+1}\right)$ where $\mathcal{H}$ is the computation graph of $\Gamma(\boldsymbol{X}, \boldsymbol{\mathcal{N}}_{\mathbb{P},Y})$ and $\Gamma(\boldsymbol{X}, \boldsymbol{\mathcal{N}}_{\mathbb{Q},Y})$.

Part II

Applications of Computing Divergences between Graphical Models

Chapter 6

Generating High-Dimensional Data with Concept Drift of Known Magnitudes

The applications of the ability to compute divergences between high-dimensional discrete distributions is vast, especially in the field of concept drift. For instance, we can learn decomposable models from data and use the divergence computation techniques developed to estimate the divergence between samples. This has wide applications such as in the problem of mapping concept drift in a dataset (Webb et al. 2018). However, in this chapter we will explore the application of divergence computation between DMs to the problem of generating synthetic datasets with occurrences of concept drift with known drift magnitudes.

Specifically, in this chapter, we will propose a method for modifying the parameters of an existing decomposable model, $\mathbb{P}_{\mathcal{G}}$, to obtain a new model $\mathbb{Q}_{\mathcal{G}}$, that is some target amount of divergence away from $\mathbb{P}_{\mathcal{G}}$. We can then sample data from these 2 models to obtain datasets with known magnitudes of concept drift.

However, when generating datasets for the purposes of testing how different models and adaptation techniques behave in the presence of concept drift, it is important to control the general "difficulty" throughout the dataset. One reason why this is important is, if the distribution after concept drift is too easy, then all the learners being tested might adapt to the new distribution too quickly, making it hard to differentiate between the adaptation performance of the various learners or adaptation techniques being tested.

Therefore, in Section 6.1, we will develop a general method for modifying any discrete distribution to ensure any resulting distribution from the modification has the same entropy as the original distribution. Once this method of modifying discrete distributions while maintaining the same entropy is established, in Section 6.2, we will apply this method to modifying the discrete distributions expressed by DMs such that the resulting modified DM is a target amount of divergence away from

the original. However, here we assumed we already have a DM to modify in the first place. Therefore, in Section 6.3, we will develop a method to generate random DMs with a limit on the treewidth of these models.

6.1 Entropy and its Relation to Dataset Difficulty

Recall that the main goal of developing a drift generator is to test how different drift adaptation techniques react to varying magnitudes of concept drift. However, it is not guaranteed for the performance and behaviour of a model on differing datasets to be similar. This can make comparing the performance and behaviour of a model or drift adaptation technique across the datasets generated with drift generator using varying *drift magnitude*s difficult.

Of course, we can try to rectify this problem by taking into account the maximum accuracy of the model on just the data before and after the occurrence of concept drift separately and using it as the baseline to compare the accuracy of the model or adaptation technique to when adapting to drift (Shaker & Hüllermeier, 2015). However, this solution can only go so far, as when trying to induce large magnitudes of concept drift in a dataset, it is likely that we will generate post-drift distributions that are easy to learn. This can result in datasets, especially high drift magnitude datasets, where the model learns the new concept too quickly, causing the different models and adaptation techniques to achieve very high and indistinguishable accuracy.

To illustrate this problem, let us consider the problem of modifying the probability vector $P = [p_1, \dots, p_k]$ where the entries of P are multiples of $1/a, a \in \mathbb{N}$, i.e. P is a "quantum distribution" with quantum $1/a$. Then we know that the distribution $Q = [q_1, \dots, q_k]$ with the maximum KL-divergence from P is the distribution where as much quanta as possible are concentrated in the entry q_i where p_i is the entry in P with the minimum value (Bonnici, 2020). More formally, let

$$P = [p_1, \dots, p_k] \quad \text{where} \quad \forall p \in P : p \cdot a \in \mathbb{N}$$

$$Q = [q_1, \dots, q_k] \quad \text{where} \quad q_i = \begin{cases} \frac{a-(k-1)}{a} & i = \arg\min_{j \in \{1,2,\dots,k\}} p_j \\ 1/a & \text{otherwise} \end{cases}$$

then we know from Bonnici (2020) that

$$Q = \arg\max_{\hat{P} \in \mathcal{S}} D_{KL}(P, \hat{P})$$

Table 6.1 provides some examples of probability vectors and the distribution with the greatest KL-divergence from them. From this example, we can observe that if

x	$\bar{P}$	$\bar{Q}$	$\tilde{P}$	$\tilde{Q}$
0	2/8	1/8	2/8	1/8
1	1/8	1/8	2/8	1/8
2	4/8	1/8	2/8	1/8
3	1/8	5/8	2/8	5/8

Table 6.1: 2 quantum distributions, $\bar{P}$ and $\tilde{P}$, with a "quantum" of 1/8. $\bar{Q}$ and $\tilde{Q}$ are the distributions with maximum KL divergence from them based on Bonnici (2020).

we were to try to modify discrete distributions to induce high magnitudes of drift, we would invariably obtain distributions that are very easy to model.

We ought to prevent this problem of generating datasets that are too "easy" at the source by ensuring that the distributions before and after the occurrence of concept drift have the same "difficulty". One way to quantify the "difficulty" of a distribution is with the Shannon entropy.

$$H(P) = -\sum_{\boldsymbol{x} \in \mathcal{X}} P_X(\boldsymbol{x}) \log P_X(\boldsymbol{x})$$

Intuitively, using the Shannon entropy to represent the difficulty of a distribution P makes sense, as entropy is a measure of the expected amount of information content in a sample from P (MacKay, 2003). Therefore, for the rest of this section we will focus on modifying the parameters of a given decomposable model, such that the modified model is a target amount of divergence away from the original DM, while still having the same entropy as the original model.

Problem 3 (Drifting DMs) Assume we are given a decomposable model $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$, a divergence to use D, and a real value d in the range of D, $d \in \text{Range}(D)$. Then the *DM drifting* problem involves modifying the parameters of $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$ such that the following are true for the distributions $\mathbb{P}$ and $\mathbb{Q}$ modelled by DMs $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$ and $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$:

$$H(\mathbb{P}) - H(\mathbb{Q}) = 0$$
$$D(\mathbb{P} \| \mathbb{Q}) = d$$

Of course, even if we did not control for entropy, it is possible for the provided target drift magnitude d to be higher than the maximum possible divergence that can be induced for some given P:

$$d > \max_{Q \in S} D(P \parallel Q)$$

where S is all the probability vectors with the same length as P. For instance, in the example distributions $\bar{P}$ and $\tilde{P}$ in Table 6.1, the maximum KL-divergence that can be induced for $\tilde{P}$ is less than $\bar{P}$:

$$\begin{aligned} D_{KL}(\bar{P} \| \bar{Q}) &= \frac{2}{8}\log(2) + \frac{1}{8}\log(1) + \frac{4}{8}\log(4) + \frac{2}{8}\log\left(\frac{1}{5}\right) \\ &= 0.6695 \\ D_{KL}(\tilde{P} \| \tilde{Q}) &= 3\frac{2}{8}\log(2) + \frac{2}{8}\log\left(\frac{2}{5}\right) \\ &= 0.4195 \\ &< D_{KL}(\bar{P} \| \bar{Q}) \end{aligned}$$

Therefore, the maximum possible divergence that can be induced depends on the initial distribution P itself. Consequently, in order to approach Problem 3, we need to be able to complete two tasks on the simpler problem of modifying a probability vector P:

1. Developing a method to modify P such that entropy remains the same.
2. Finding the probability vector with the largest possible divergence away from P that is obtainable with the method developed for maintaining entropy.

We then need to apply this method for modifying probability vectors to modifying the parameters of the given decomposable model.

6.1.1 Permuting Discrete Distributions and Maintaining Entropy

In this section we will tackle the problem of modifying the parameters of a given DM such that the resulting DM is some target divergence away from the original model. To better explain how we will tackle this problem, we will first discuss the same problem but on a given probability vector, instead of a DM. We will then show how our problem of modifying the parameters of a given DM to achieve some target divergence in both the joint and conditional distribution is similar to this easier problem of modifying a single probability vector for the same goal.

Observe from the definition of Shannon entropy of a discrete distribution P

$$H(P) = -\sum_{x \in \mathcal{X}} P_X(\boldsymbol{x}) \log P_X(\boldsymbol{x})$$

that P and any permutation of P will have the same entropy as the order of the entries in the distribution does not matter when computing entropy.

Proposition 10 (Permutations maintain entropy) Let $P = [p_1, \ldots, p_k]$ be a probability vector. If a probability vector Q is a permutation of P, then their Shannon entropy are the same, $H(P) = H(Q)$.

Proof Since Q is a permutation of P, there is a mapping π

$$\begin{aligned} & \pi : \{1, \ldots, k\} \to \{1, \ldots, k\} \\ \text{s.t.} \quad & \forall i \in \{1, \ldots, k\} : P[i] = Q[\pi(i)] \end{aligned}$$

which implies

$$H(P) - H(Q) = \sum_{i \in \{1,\dots,k\}} P[i] - \sum_{i \in \{1,\dots,k\}} Q[i] = \sum_{i \in \{1,\dots,k\}} P[i] - Q[\pi(i)] = 0$$

and therefore P and any permutation of P, Q, have the same Shannon entropy.

However, Proposition 10 only proves that this implication goes in one direction, it might be possible to find a modification of P, Q, that is both not a permutation of P but still have the same entropy as P. Therefore, by only considering permutations of P when finding a distribution with a given divergence away from P, we are unfortunately restricting ourselves to a subset of all possible distributions that has the same entropy as P. However, this restriction still results in a search space that grows factorially with respect to the size of P. Knowing that permutations of probability vectors maintain the entropy of the distribution, the question then is: how do we apply this fact and restriction to modifying the parameters of a DM in order to achieve some target divergence in the joint or conditional distribution represented by the DM?

6.1.1.1 Joint

In order to approach the problem of inducing some amount of drift in the joint distribution of a DM while maintaining the same entropy, first recall the Shannon entropy of the joint distribution $\mathbb{P}$ modelled by the DM $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$

$$\begin{aligned}
H(\mathbb{P}) &= \sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{P}(\boldsymbol{x}) \log \mathbb{P}(\boldsymbol{x}) \\
&= \sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{P}(\boldsymbol{x}) \log \frac{\prod_{C \in \boldsymbol{C}} \mathbb{P}_C(\boldsymbol{x}_C)}{\prod_{S \in \boldsymbol{S}} \mathbb{P}_S(\boldsymbol{x}_S)} \\
&= \sum_{C \in \boldsymbol{C}} \sum_{\boldsymbol{x}_C \in \mathcal{X}_C} \log \mathbb{P}_C(\boldsymbol{x}_C) \sum_{\boldsymbol{x} \in \mathcal{X}_{X-C}} \mathbb{P}(\boldsymbol{x}_C, \boldsymbol{x}) - \sum_{S \in \boldsymbol{S}} \sum_{\boldsymbol{x}_S \in \mathcal{X}_S} \log \mathbb{P}_S(\boldsymbol{x}_S) \sum_{\boldsymbol{x} \in \mathcal{X}_{X-S}} \mathbb{P}(\boldsymbol{x}_S, \boldsymbol{x}) \\
&= \sum_{C \in \boldsymbol{C}} \sum_{\boldsymbol{x}_C \in \mathcal{X}_C} \mathbb{P}_C(\boldsymbol{x}_C) \log \mathbb{P}_C(\boldsymbol{x}_C) - \sum_{S \in \boldsymbol{S}} \sum_{\boldsymbol{x}_S \in \mathcal{X}_S} \mathbb{P}_S(\boldsymbol{x}_S) \log \mathbb{P}_S(\boldsymbol{x}_S) \\
&= \sum_{C \in \boldsymbol{C}} H(\mathbb{P}_C) - \sum_{S \in \boldsymbol{S}} H(\mathbb{P}_S)
\end{aligned}$$

From this we can observe that modifying the parameters in a DM such that entropy remains the same is not as simple as permuting the clique probabilities of some maximal clique in $\boldsymbol{C}$. This is due to modifications in clique probabilities causing both changes in the entropy of separator probabilities and also inconsistencies in the minimal separator probabilities with adjacent maximal cliques. One way to avoid this issue is to only modify the probabilities over the variables in the maximal clique that are not shared with any other maximal cliques in the DM, $\forall C \in \boldsymbol{C} \,:\, C - \bigcup(\boldsymbol{C} \setminus \{C\})$.

Theorem 13 (Permuting DMs while maintaining joint entropy) For any decomposable model $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$ with maximal cliques $\boldsymbol{C}$ and minimal separators $\boldsymbol{S}$, let $\circ C$ and $\bullet C$ be the vertices in C that are shared with other maximal cliques and that are exclusive in C respectively.

$$\circ C = C \bigcap \bigcup \boldsymbol{S}$$
$$\bullet C = C \setminus \circ C$$

Then permuting the rows of the CPTs

$$\forall C \in \boldsymbol{C}(\mathbb{P}_{\mathcal{G}_\mathbb{P}}) : \frac{\mathbb{P}_C}{\sum_{\boldsymbol{x}_{\cdot C} \in \mathcal{X}_{\cdot C}} \mathbb{P}_C(\boldsymbol{x}_{\cdot C})} = \frac{\mathbb{P}_C}{\mathbb{P}_{\circ C}} = \mathbb{P}_{\cdot C|\circ C}$$

will ensure that the resulting decomposable model with these modifications, $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$, will have the same entropy, $H(\mathbb{P}) = H(\mathbb{Q})$.

Proof Let $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$ be the DM after permuting the rows

$$\forall C \in \boldsymbol{C}(\mathbb{P}_{\mathcal{G}_\mathbb{P}}) : \mathbb{P}_{\cdot C|\circ C}$$

resulting in the following joint probabilities over the maximal cliques in $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$:

$$\forall C \in \boldsymbol{C} : \mathbb{Q}_C = \mathbb{Q}_{\bullet C \mid \circ C}\mathbb{P}_{\circ C}$$

where

$$\boldsymbol{C} = \boldsymbol{C}(\mathbb{P}_{\mathcal{G}_\mathbb{P}}) = \boldsymbol{C}(\mathbb{Q}_{\mathcal{G}_\mathbb{Q}})$$
$$\forall C \in \boldsymbol{C} : \mathbb{Q}_{\circ C} = \mathbb{P}_{\circ C}$$

and $\mathbb{Q}_{\cdot C|\circ C}$ is the CPT $\mathbb{P}_{\cdot C|\circ C}$ after carrying out some set of permutations on its rows. In order for the entropy of $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$ and $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$ to be the same we require the following to be true:

$$\begin{aligned} 0 &= H(\mathbb{P}) - H(\mathbb{Q}) \\ &= \sum_{C \in \boldsymbol{C}} H(\mathbb{P}_C) - H(\mathbb{Q}_C) - \sum_{S \in \boldsymbol{S}} H(\mathbb{P}_S) - H(\mathbb{Q}_S) \end{aligned}$$

Therefore by showing that

$$\forall C \in \boldsymbol{C} : H(\mathbb{P}_C) = H(\mathbb{Q}_C)$$
$$\forall S \in \boldsymbol{S} : H(\mathbb{P}_S) = H(\mathbb{Q}_S)$$

we will show that $H(\mathbb{P}) = H(\mathbb{Q})$. Showing $H(\mathbb{P}_S) = H(\mathbb{Q}_S)$ can be done directly as:

$$\begin{aligned}
&\forall C \in \boldsymbol{C} : \mathbb{P}_{\circ C} = \mathbb{Q}_{\circ C} \\
&\Rightarrow \forall S \in \boldsymbol{S} : \mathbb{P}_S = \mathbb{Q}_S \\
&\Rightarrow \forall S \in \boldsymbol{S} : H(\mathbb{P}_S) = H(\mathbb{Q}_S)
\end{aligned}$$

Showing that $H(\mathbb{P}_C) = H(\mathbb{Q}_C)$ on the other hand is a bit more involved. First observe that $H(\mathbb{P}_C)$ can be decomposed into a weighted sum of the entropy of $\mathbb{P}_{\bullet C|\circ C}$ and the entropy of $\mathbb{P}_{\circ C}$:

$$\begin{aligned}
&H(\mathbb{P}_C) \\
&= \sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{P}_{\bullet C|\circ C}(\boldsymbol{x}) \mathbb{P}_{\circ C}(\boldsymbol{x}) \log \mathbb{P}_{\bullet C|\circ C}(\boldsymbol{x}) \mathbb{P}_{\circ C}(\boldsymbol{x}) \\
&= \sum_{\boldsymbol{x}_{\circ C} \in \mathcal{X}_{\circ C}} \mathbb{P}_{\circ C}(\boldsymbol{x}_{\circ C}) \sum_{\boldsymbol{x}_{\bullet C} \in \mathcal{X}_{\bullet C}} \mathbb{P}_{\bullet C|\circ C}(\boldsymbol{x}_{\bullet C} \mid \boldsymbol{x}_{\circ C}) \log \mathbb{P}_{\bullet C|\circ C}(\boldsymbol{x}_{\bullet C} \mid \boldsymbol{x}_{\circ C}) + \\
&\quad \sum_{\boldsymbol{x}_{\circ C} \in \mathcal{X}_{\circ C}} \mathbb{P}_{\circ C}(\boldsymbol{x}_{\circ C}) \log \mathbb{P}_{\circ C}(\boldsymbol{x}_{\circ C}) \sum_{\boldsymbol{x}_{\bullet C} \in \mathcal{X}_{\bullet C}} \mathbb{P}_{\bullet C|\circ C}(\boldsymbol{x}_{\bullet C} \mid \boldsymbol{x}_{\circ C}) \\
&= \sum_{\boldsymbol{x}_{\circ C} \in \mathcal{X}_{\circ C}} \mathbb{P}_{\circ C}(\boldsymbol{x}_{\circ C}) H(\mathbb{P}_{\bullet C|\circ C}) + H(\mathbb{P}_{\bullet C})
\end{aligned}$$

Then recall that permuting the CPT row $\mathbb{P}_{X_{\bullet C}|\boldsymbol{x}_{\circ C}}$ ensures that its entropy remains the same. This implies that

$$\forall C \in \boldsymbol{C}, \forall \boldsymbol{x}_{\circ C} \in \mathcal{X}_{\circ C} : H(\mathbb{P}_{X_{\bullet C}|\boldsymbol{x}_{\circ C}}) = H(\mathbb{Q}_{X_{\bullet C}|\boldsymbol{x}_{\circ C}})$$

Then taking the difference between $H(\mathbb{P}_C)$ and $H(\mathbb{Q}_C)$:

$$\begin{aligned}
&\forall C \in \boldsymbol{C} : \\
&\quad H(\mathbb{P}_C) - H(\mathbb{Q}_C) \\
&\quad = \sum_{\boldsymbol{x}_{\circ C} \in \mathcal{X}_{\circ C}} \mathbb{P}_{\circ C}(\boldsymbol{x}_{\circ C}) H(\mathbb{P}_{\bullet C|\circ C}) + H(\mathbb{P}_{\bullet C}) - \sum_{\boldsymbol{x}_{\circ C} \in \mathcal{X}_{\circ C}} \mathbb{P}_{\circ C}(\boldsymbol{x}_{\circ C}) H(\mathbb{Q}_{\bullet C|\circ C}) - H(\mathbb{Q}_{\bullet C}) \\
&\quad = \sum_{\boldsymbol{x}_{\circ C} \in \mathcal{X}_{\circ C}} \mathbb{P}_{\circ C}(\boldsymbol{x}_{\circ C}) H(\mathbb{P}_{\bullet C|\circ C}) - \mathbb{P}_{\circ C}(\boldsymbol{x}_{\circ C}) H(\mathbb{Q}_{\bullet C|\circ C}) \\
&\quad = \sum_{\boldsymbol{x}_{\circ C} \in \mathcal{X}_{\circ C}} \mathbb{P}_{\circ C}(\boldsymbol{x}_{\circ C}) \cdot 0 \\
&\quad = 0
\end{aligned}$$

Therefore, for all $C \in \boldsymbol{C}$, $H(\mathbb{P}_C) = H(\mathbb{Q}_C)$, and since we already know $H(\mathbb{P}_S) = H(\mathbb{Q}_S)$ for all $S \in \boldsymbol{S}$, we can conclude that $H(\mathbb{P}) = H(\mathbb{Q})$ when only row in the CPTs $\mathbb{P}_{\bullet C|\circ C}$ for some DM is permuted.

Therefore, the full set of CPT rows that we can permute to induce some amount of drift in a DM while preserving the same entropy is:

$$\left\{ \mathbb{P}_{\boldsymbol{X}_{\bullet C}|\boldsymbol{x}_{\circ C}} \;\middle|\; \forall \boldsymbol{C} \in \boldsymbol{C}, \forall \boldsymbol{x}_{\circ C} \in \mathcal{X}_{\circ C} \right\}. \tag{6.1}$$

6.1.1.2 Conditional

In order to approach the problem of inducing some amount of drift in the conditional distribution of a DM while maintaining the same entropy, first recall the Shannon entropy of the conditional distribution $\mathbb{P}$ modelled by the DM $\mathbb{P}_{\mathcal{G}_{\mathbb{P}}}$

$$\begin{aligned} H(\mathbb{P}_{Y|Z}) &= \mathbb{E}_Z[H(\mathbb{P}_{Y|z})] \\ &= \sum_{z \in \mathcal{Z}} \mathbb{P}_Z(\boldsymbol{z}) H(\mathbb{P}_{Y|z}) \end{aligned}$$

Using the definition of $\boldsymbol{\mathcal{B}}$-partitions from Definition 23 and the decomposition of conditional distributions over DMs from Section 5.1

$$\begin{aligned} \forall \boldsymbol{z} \in \mathcal{Z} : & H(\mathbb{P}_{Y|z}) \\ &= \sum_{\boldsymbol{y} \in \mathcal{Y}} \mathbb{P}_{Y|z}(\boldsymbol{y}) \log \mathbb{P}_{Y|z}(\boldsymbol{y}) \\ &= \sum_{\boldsymbol{y} \in \mathcal{Y}} \prod_{N \in \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathbb{P},Y})} \mathbb{P}_{Y_N|z_N}(\boldsymbol{y}_N) \log \prod_{N \in \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathbb{P},Y})} \mathbb{P}_{Y_N|z_N}(\boldsymbol{y}_N) \\ &= \sum_{N \in \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathbb{P},Y})} \sum_{\boldsymbol{y} \in \mathcal{Y}} \prod_{N' \in \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathbb{P},Y})} \mathbb{P}_{Y_{N'}|z_{N'}}(\boldsymbol{y}_{N'}) \log \mathbb{P}_{Y_N|z_N}(\boldsymbol{y}_N) \\ &= \sum_{N \in \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathbb{P},Y})} \sum_{\boldsymbol{y}_N \in \mathcal{Y}_N} \mathbb{P}_{Y_N|z_N}(\boldsymbol{y}_N) \log \mathbb{P}_{Y_N|z_N}(\boldsymbol{y}_N) \prod_{N' \in \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathbb{P},Y}) \setminus \{N\}} \sum_{\boldsymbol{y}_{N'} \in \mathcal{Y}_{N'}} \mathbb{P}_{Y_{N'}|z_{N'}}(\boldsymbol{y}_{N'}) \\ &= \sum_{N \in \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathbb{P},Y})} \sum_{\boldsymbol{y}_N \in \mathcal{Y}_N} \mathbb{P}_{Y_N|z_N}(\boldsymbol{y}_N) \log \mathbb{P}_{Y_N|z_N}(\boldsymbol{y}_N) \prod_{N' \in \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathbb{P},Y}) \setminus \{N\}} 1 \\ &= \sum_{N \in \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathbb{P},Y})} \sum_{\boldsymbol{y}_N \in \mathcal{Y}_N} \mathbb{P}_{Y_N|z_N}(\boldsymbol{y}_N) \log \mathbb{P}_{Y_N|z_N}(\boldsymbol{y}_N) \\ &= \sum_{N \in \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathbb{P},Y})} H(\mathbb{P}_{Y_N|z_N}) \end{aligned}$$

Since $\mathbb{P}_{Y_N|z_N}$ are probability vectors, by Proposition 10, $H(\mathbb{P}_{Y_N|z_N})$ is invariant under permutations of $\mathbb{P}_{Y_N|z_N}$. Furthermore, the equivalence $H(\mathbb{P}_{Y|z}) = \sum_{N \in \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathbb{P},Y})} H(\mathbb{P}_{Y_N|z_N})$ implies the following are invariant to permutations of $\mathbb{P}_{Y_N|z_N}$:

$$\Rightarrow H(\mathbb{P}(\boldsymbol{Y} \mid \boldsymbol{z}))$$

$$\Rightarrow \textstyle\sum_{z \in \mathcal{Z}} \mathbb{P}_Z(\boldsymbol{z}) H(\mathbb{P}(\boldsymbol{Y} \mid \boldsymbol{z}))$$

$$\Rightarrow H(\mathbb{P}(\boldsymbol{Y} \mid \boldsymbol{Z}))$$

Therefore, permutations of the following set of CPT rows will preserve the conditional entropy $H(\mathbb{P}_{Y|Z})$ of the DM being modified:

$$\left\{ \mathbb{P}_{Y_N|z_N} \;\middle|\; N \in \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{N}}_{\mathbb{P},Y}), z_N \in \mathcal{Z}_N \right\} \tag{6.2}$$

6.1.2 Distribution with Maximum Divergence

Now that we have shown a method for modifying the parameters of a DM to achieve some target divergence in either the joint or conditional distribution such that its entropy remains the same, we now need to tackle the problem of finding the maximum amount of divergence that can be induced via this method. Since, fundamentally, the method of modifying the parameters in a DM described in Section 6.1.1 revolves around permuting the rows of appropriate CPTs, which are just individual probability vectors, we will first consider the problem of finding the permutation of a probability vector P with the largest $\alpha\beta$-divergence away from P.

When tackling this problem, we can assume, without loss of generality, that P is a probability vector that has entries in ascending order,

$$P = [p_1, \ldots, p_k] \quad s.t. \quad \forall i < j \,:\, p_i < p_j,$$

as, for any unsorted vector $\bar{P}$, we can first sort $\bar{P}$ to obtain P, find the vector Q with the greatest possible $\alpha\beta$-divergence from P, then find the vector $\bar{Q}$ that has the greatest $\alpha\beta$-divergence from $\bar{P}$ by applying, on Q, the inverse of the ordering used to obtain P from $\bar{P}$.

Theorem 14 (Permutation of with maximum $\alpha\beta$-divergence) Assume, without loss of generality, that P is a probability vector that is sorted in ascending order:

$$P = [p_1, \ldots, p_k] \quad s.t. \quad \forall i < j \,:\, p_i < p_j$$

Then the permutation of P with the greatest $\alpha\beta$-divergence from P is the probability vector with the reverse order of P:

$$\arg\max_{Q \in \mathcal{Q}} D_{\alpha\beta}(P \| Q) = [p_k, \ldots, p_1]$$

where $\mathcal{Q}$ is the set of all possible permutations of P.

Proof First recall that the definition of the extended $\alpha\beta$-divergence is divided into multiple cases:

$$D_{AB}^{\alpha,\beta}(\mathbb{P},\mathbb{Q}) = \sum_{\boldsymbol{x}\in\mathcal{X}} d_{AB}^{\alpha,\beta}\big(\mathbb{P}(\boldsymbol{x}),\mathbb{Q}(\boldsymbol{x})\big)$$

where

$$d_{AB}^{(\alpha,\beta)}(\mathbb{P}(\vec{x}),\mathbb{Q}(\vec{x})) = \begin{cases} -\frac{1}{\alpha\beta}\left(\mathbb{P}(\vec{x})^{\alpha}\mathbb{Q}(\vec{x})^{\beta} - \frac{\alpha\mathbb{P}(\vec{x})^{\alpha+\beta}}{\alpha+\beta} - \frac{\beta\mathbb{Q}(\vec{x})^{\alpha+\beta}}{\alpha+\beta}\right) & \text{for } \alpha,\beta,\alpha+\beta\neq 0 \\ \frac{1}{\alpha^2}\left(\mathbb{P}(\vec{x})^{\alpha}\log\frac{\mathbb{P}(\vec{x})^{\alpha}}{\mathbb{Q}(\vec{x})^{\alpha}} - \mathbb{P}(\vec{x})^{\alpha} + \mathbb{Q}(\vec{x})^{\alpha}\right) & \text{for } \alpha\neq 0,\beta=0 \\ \frac{1}{\alpha^2}\left(\log\frac{\mathbb{Q}(\vec{x})^{\alpha}}{\mathbb{P}(\vec{x})^{\alpha}} + \left(\frac{\mathbb{Q}(\vec{x})^{\alpha}}{\mathbb{P}(\vec{x})^{\alpha}}\right)^{-1} - 1\right) & \text{for } \alpha=-\beta\neq 0 \\ \frac{1}{\beta^2}\left(\mathbb{Q}(\vec{x})^{\beta}\log\frac{\mathbb{Q}(\vec{x})^{\beta}}{\mathbb{P}(\vec{x})^{\beta}} - \mathbb{Q}(\vec{x})^{\beta} + \mathbb{P}(\vec{x})^{\beta}\right) & \text{for } \alpha=0,\beta\neq 0 \\ \frac{1}{2}(\log\mathbb{P}(\vec{x}) - \log\mathbb{Q}(\vec{x}))^2 & \text{for } \alpha,\beta=0. \end{cases} \tag{2.10}$$

Therefore, we will approach proving Theorem 14 by proving it for the 5 cases in Equation (2.10). These proofs will involve first assuming the existence of 2 permutations of P that are different by a single swap:

$$\bar{Q} = [\ldots, q_i, \ldots, q_j, \ldots]$$
$$\tilde{Q} = [\ldots, q_j, \ldots, q_i, \ldots]$$

where:

$$q_i < q_j$$

as in the vector $\bar{Q}$ is a vector with entries that are not sorted in descending order and vector $\tilde{Q}$ is obtained by swapping these entries. Then by showing that the difference in divergence from P to $\bar{Q}$ and from P to $\tilde{Q}$ is negative,

$$D_{\alpha\beta}^{(\alpha,\beta)}(P \parallel \bar{Q}) - D_{\alpha\beta}^{(\alpha,\beta)}(P \parallel \tilde{Q}) \leq 0 \Rightarrow D_{\alpha\beta}^{(\alpha,\beta)}(P \parallel \bar{Q}) \leq D_{\alpha\beta}^{(\alpha,\beta)}(P \parallel \tilde{Q})$$

we will show that for any vector $\bar{Q}$ whose entries are not sorted in descending order, we can construct a new vector $\tilde{Q}$ that will have a greater $\alpha\beta$-divergence from P. Conversely, the only vector $\bar{Q}$ with no $\tilde{Q}$ counterpart, and therefore no vector with a greater $\alpha\beta$-divergence from P, is the vector with entries in descending order, i.e. the vector with the reverse order of P.

With the rough argument of this proof outlined, we will now show that for all values of α and β, $D_{\alpha\beta}^{(\alpha,\beta)}(P \parallel \bar{Q}) - D_{\alpha\beta}^{(\alpha,\beta)}(P \parallel \tilde{Q}) \leq 0$ and therefore the permutation of P with the greatest $\alpha\beta$-divergence from P, is P but in reverse order.

when $\alpha, \beta, \alpha+\beta \neq 0$

$$
\begin{aligned}
&D_{\alpha\beta}^{(\alpha,\beta)}(p \parallel \bar{q}) - D_{\alpha\beta}^{(\alpha,\beta)}(p \parallel \tilde{q}) \\
&= \sum_{l=1}^{k} -\frac{1}{\alpha\beta}\left(p_l^\alpha \bar{q}_i^\beta - \frac{\alpha p_i^{\alpha+\beta}}{\alpha+\beta} - \frac{\beta \bar{q}_l^{\alpha+\beta}}{\alpha+\beta} \right) + \frac{1}{\alpha\beta}\left(p_l^\alpha \tilde{q}_i^\beta - \frac{\alpha p_i^{\alpha+\beta}}{\alpha+\beta} - \frac{\beta \tilde{q}_i^{\alpha+\beta}}{\alpha+\beta} \right) \\
&= \frac{1}{\alpha\beta} \sum_{l=1}^{k} p_l^\alpha \tilde{q}_i^\beta - p_l^\alpha \bar{q}_i^\beta \\
&= \frac{1}{\alpha\beta}\left(p_i^\alpha \cdot q_j^\beta - p_i^\alpha \cdot q_i^\beta + p_j^\alpha \cdot q_i^\beta - p_j^\alpha \cdot q_j^\beta \right) \\
&= \frac{1}{\alpha\beta}\left(p_i^\alpha (q_j^\beta - q_i^\beta) - p_j^\alpha (q_j^\beta - q_i^\beta) \right) \\
&= \frac{1}{\alpha\beta}\left((p_i^\alpha - p_j^\alpha) \cdot (q_j^\beta - q_i^\beta) \right) \\
&\leq 0 \quad \left(\text{since } p_i^\alpha - p_j^\alpha \leq 0 \,,\, q_j^\beta - q_i^\beta \geq 0 \text{ and } 1/(\alpha\beta) > 0 \right)
\end{aligned}
$$

when $\alpha \neq 0, \beta = 0$

$$
\begin{aligned}
&D_{\alpha\beta}^{(\alpha,\beta)}(p \parallel \bar{q}) - D_{\alpha\beta}^{(\alpha,\beta)}(p \parallel \tilde{q}) \\
&= \sum_{l=1}^{k} \frac{1}{\alpha^2}\left(p_l^\alpha \log \frac{p_l^\alpha}{\bar{q}_l^\alpha} - p_l^\alpha + \bar{q}_l^\alpha \right) - \frac{1}{\alpha^2}\left(p_l^\alpha \log \frac{p_l^\alpha}{\tilde{q}_l^\alpha} - p_l^\alpha + \tilde{q}_l^\alpha \right) \\
&= \frac{1}{\alpha^2} \sum_{l=1}^{k}\left(p_l^\alpha \log \frac{p_l^\alpha}{\bar{q}_l^\alpha} - p_l^\alpha \log \frac{p_l^\alpha}{\tilde{q}_l^\alpha} \right) \\
&= \frac{1}{\alpha^2}\left(p_i^\alpha \log \frac{p_i^\alpha}{q_i^\alpha} - p_i^\alpha \log \frac{p_i^\alpha}{q_j^\alpha} + p_j^\alpha \log \frac{p_j^\alpha}{q_j^\alpha} - p_j^\alpha \log \frac{p_j^\alpha}{q_i^\alpha} \right) \\
&= \frac{1}{\alpha^2}\left(p_i^\alpha \log \frac{q_j^\alpha}{q_i^\alpha} - p_j^\alpha \log \frac{q_j^\alpha}{q_i^\alpha} \right) \\
&= \frac{1}{\alpha^2}\left(\left(p_i^\alpha - p_j^\alpha \right) \log \frac{q_j^\alpha}{q_i^\alpha} \right) \\
&\leq 0 \quad \left(\text{since } q_j \geq q_i \Rightarrow \frac{q_j}{q_i} \geq 1 \Rightarrow \alpha \cdot \log \frac{q_j}{q_i} \geq 0 \text{ and } p_i^\alpha - p_j^\alpha \leq 0 \right)
\end{aligned}
$$

when $\alpha = -\beta \neq 0$

$$
D_{\alpha\beta}^{(\alpha,\beta)}(p \parallel \bar{q}) - D_{\alpha\beta}^{(\alpha,\beta)}(p \parallel \tilde{q})
$$

$$
\begin{aligned}
&= \sum_{l=1}^{k} \frac{1}{\alpha^2}\left(\log \frac{\bar{q}_l^{\alpha}}{p_l^{\alpha}} + \left(\frac{\bar{q}_l^{\alpha}}{p_l^{\alpha}}\right)^{-1} - 1\right) - \frac{1}{\alpha^2}\left(\log \frac{\tilde{q}_l^{\alpha}}{p_l^{\alpha}} + \left(\frac{\tilde{q}_l^{\alpha}}{p_l^{\alpha}}\right)^{-1} - 1\right) \\
&= \frac{1}{\alpha^2}\sum_{l=1}^{k}\left(\log \frac{\bar{q}_l^{\alpha}}{\tilde{q}_l^{\alpha}} + \frac{p_l^{\alpha}}{\bar{q}_l^{\alpha}} - \frac{p_l^{\alpha}}{\tilde{q}_l^{\alpha}}\right) \\
&= \frac{1}{\alpha^2}\left(\log \frac{q_i^{\alpha}}{q_j^{\alpha}} + \frac{p_i^{\alpha}}{q_i^{\alpha}} - \frac{p_i^{\alpha}}{q_j^{\alpha}} + \log \frac{q_j^{\alpha}}{q_i^{\alpha}} + \frac{p_j^{\alpha}}{q_j^{\alpha}} - \frac{p_j^{\alpha}}{q_i^{\alpha}}\right) \\
&= \frac{1}{\alpha^2}\left(p_i^{\alpha}\left(1/q_i^{\alpha} - q_j^{\alpha}\right) - p_j^{\alpha}\left(1/q_i^{\alpha} - q_j^{\alpha}\right)\right) \\
&= \frac{1}{\alpha^2}\left(\left(p_i^{\alpha} - p_j^{\alpha}\right)\left(1/q_i^{\alpha} - q_j^{\alpha}\right)\right) \\
&\leq 0 \quad \left(\text{since } q_i \leq q_j \Rightarrow 1/q_i^{\alpha} - 1/q_j^{\alpha} \geq 0 \text{ and } p_i^{\alpha} - p_j^{\alpha} \leq 0\right)
\end{aligned}
$$

when $\alpha = 0, \beta \neq 0$

$$
\begin{aligned}
&D_{\alpha\beta}^{(\alpha,\beta)}(p \parallel \bar{q}) - D_{\alpha\beta}^{(\alpha,\beta)}(p \parallel \tilde{q}) \\
&= \sum_{l=1}^{k} \frac{1}{\beta^2}\left(\bar{q}_l^{\beta} \log \frac{\bar{q}_l^{\beta}}{p_l^{\beta}} - \bar{q}_l^{\beta} + p_l^{\beta}\right) - \frac{1}{\beta^2}\left(\tilde{q}_l^{\beta} \log \frac{\tilde{q}_l^{\beta}}{p_l^{\beta}} - \tilde{q}_l^{\beta} + p_l^{\beta}\right) \\
&= \frac{1}{\beta^2}\sum_{l=1}^{k}\left(\bar{q}_l^{\beta} \log \frac{\bar{q}_l^{\beta}}{p_l^{\beta}} - \tilde{q}_l^{\beta} \log \frac{\tilde{q}_l^{\beta}}{p_l^{\beta}}\right) \\
&= \frac{1}{\beta^2}\left(q_i^{\beta} \log \frac{q_i^{\beta}}{p_i^{\beta}} - q_j^{\beta} \log \frac{q_j^{\beta}}{p_i^{\beta}} + q_j^{\beta} \log \frac{q_j^{\beta}}{p_j^{\beta}} - q_i^{\beta} \log \frac{q_i^{\beta}}{p_j^{\beta}}\right) \\
&= \frac{1}{\beta^2}\left(q_i^{\beta} \log \frac{p_j^{\beta}}{p_i^{\beta}} - q_j^{\beta} \log \frac{p_j^{\beta}}{p_i^{\beta}}\right) \\
&= \frac{1}{\beta^2}\left(\left(q_i^{\beta} - q_j^{\beta}\right) \log \frac{p_j^{\beta}}{p_i^{\beta}}\right) \\
&\leq 0 \quad \left(\text{since } p_j \geq p_i \Rightarrow \frac{p_j}{p_i} \geq 1 \Rightarrow \beta \cdot \log \frac{p_j}{p_i} \geq 0 \text{ and } q_i^{\beta} - q_j^{\beta} \leq 0\right)
\end{aligned}
$$

when $\alpha, \beta = 0$

$$
\begin{aligned}
&D_{\alpha\beta}^{(0,0)}(p \parallel \bar{q}) - D_{\alpha\beta}^{(0,0)}(p \parallel \tilde{q}) \\
&= \sum_{l=1}^{k} \frac{1}{2}\left(\log p_i - \log \bar{q}_i\right)^2 - \frac{1}{2}\left(\log p_i - \log \tilde{q}_i\right)^2
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{1}{2}\sum_{l=1}^{k}\left(\log p_l\right)^2 - \left(\log p_l\right)^2 + \left(\log \bar{q}_l\right)^2 - \left(\log \tilde{q}_l\right)^2 - \\
&\qquad 2\log p_l \log \bar{q}_l + 2\log p_l \log \tilde{q}_l \\
&= \sum_{l=1}^{k}\log p_l\left(\log \tilde{q}_l/\bar{q}_l\right) \\
&= \log p_i\left(\log q_j/q_i\right) + \log p_j\left(\log q_i/q_j\right) \\
&= \left(\log p_i/p_j\right)\left(\log q_j/q_i\right) \\
&\leq 0 \quad \left(\text{since } \log p_i/p_j \leq 0 \text{ and } \log q_j/q_i \geq 0\right)
\end{aligned}
$$

We can then directly apply this fact back into drifting DMs by "reversing" the order of all the CPT rows in the sets in Equations (6.1) and (6.2) for joint and conditional drift respectively.

6.2 Drifting Parameters of a Decomposable Model

So far, we have shown that the problem of inducing both joint and conditional drift in a DM while preserving its entropy can be seen as a problem of permuting the set of CPT rows in Equations (6.1) and (6.2) respectively. Furthermore we also know the exact permutation of each CPT row that will result in a DM with the maximum amount of $\alpha\beta$-divergence away from the original DM. The question now is: how do we find the permutations of the relevant set of CPT rows such that the resulting DM, $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$, is some target $\alpha\beta$-divergence away from the original DM, $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$?

Problem 4 (Drift DM via permutations) Assume we are given a decomposable model $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$, a divergence D, a real value d in the range of D, $d \in \text{Range}(D)$, and a list of CPT rows in $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$, R, to permute. Then the *drift DM via permutations* problem involves finding permutations of the probability vectors in R, which results in a modified DM, $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$, such that the following are true for the distributions $\mathbb{P}$ and $\mathbb{Q}$ modelled by DMs $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$ and $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$:

$$
\begin{aligned}
H(\mathbb{P}) - H(\mathbb{Q}) &= 0 \\
D(\mathbb{P}\|\mathbb{Q}) &= \min(d, \text{max\_div}(D, \mathbb{P}_{\mathcal{G}_\mathbb{P}}, R)) = d^*
\end{aligned}
$$

where $\text{max\_div}(D, \mathbb{P}_{\mathcal{G}_\mathbb{P}}, R)$ is the maximum value of divergence D that can be induced by permuting the vectors in R for DM $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$. Therefore, if the given divergence target d is greater than the largest divergence achievable via permutations, we just take the latter as the new target divergence.

There are many ways to approach this problem. The approach that we will present is just a simple example of how one might tackle Problem 4. Our approach at tackling Problem 4 will involve 2 steps.

1. Coarse Drift: The first step involves roughly selecting which rows in the set of CPT rows R we want to permute by the "maximal" amount such that the resulting overall drift is above the target drift,

2. Fine Drift: the second step is to reduce the drift induced in step 1 by reducing the drift in each row that was "maximally" drifted in step 1 until the target drift is reached.

Algorithm 1: Drift Step 1: Coarse drift

Data: An initial decomposable model $\mathbb{P}_{\mathcal{G}}$, list of CPT rows to modify `rows`, and a target value d for divergence D

Result: Decomposable model $\mathbb{Q}_{\mathcal{G}}$ where $D(\mathbb{P}, \mathbb{Q}) \geqq d$, list of CPT rows modified

$\mathbb{Q}_{\mathcal{G}}$ = copy of $\mathbb{P}_{\mathcal{G}}$;
`drifted_vecs` = [];
`cur_drift` = 0;
for *row in* `rows` **do**
    Drift vector `row` in $\mathbb{Q}$ maximally;
    $c = D(\mathbb{P}, \mathbb{Q})$;
    `drifted_vecs`.push(`row`);
    `cur_drift` = c;
    if $c > d$ **then**
        Output $\mathbb{Q}$ and `drifted_vecs`;
    end
end
Output $\mathbb{Q}$ and `drifted_vecs`;

Next we present Algorithm 1 to carry out step 1 of our proposed method. The main output of this algorithm is a DM $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$ such that the divergence between it and the original DM, $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$, is above the target divergence d^*. Algorithm 1 also returns a list of CPT rows that it has modified to give as input to Algorithm 2. By ensuring that $D(\mathbb{P}, \mathbb{Q}) \geq d^*$, and providing the list of CPT rows modified to achieve this amount of drift, we provide Algorithm 2 the opportunity to reduce the drift induced in each row such that $D(\mathbb{P}, \mathbb{Q})$ is less than but close to d^*.

As a result a lot of the computational burden is theoretically placed on Algorithm 2 as it has the responsibility of searching through a factorial search space for each CPT row modified by Algorithm 1. However, instead of tackling this large search space directly, Algorithm 2 limits itself to permutations that are obtained from swapping the $\pi_r(i)$-th element of any row, r, with the $\pi_r(\text{len}(r) - 1 - i)$-th element,

Algorithm 2: Drift Step 2: Fine drift

Data: An initial decomposable model $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$, modified decomposable model from coarse drift step $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$, list of CPT rows reversed in coarse drift step `rows`, and a target value d for divergence D

Result: Decomposable model $\mathbb{Q}_\mathcal{G}$ where $D(\mathbb{P}_\mathcal{G}, \mathbb{Q}_\mathcal{G}) \leq d$

for r *in* `rows` **do**
    π_r = Permutation mapping to convert r into a sorted vector;
    for i *in* $[0, floor(len(r)/2) - 1]$ **do**
        $j = \text{len}(r) - 1 - i$;
        Swap entry $\pi_r(i)$ and $\pi_r(j)$ of row r in $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$;
        if $D(\mathbb{P}, \mathbb{Q}) < d$ **then**
            Swap entry $\pi_r(i)$ and $\pi_r(j)$ of r in $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$;
        end
    end
end
`closest_mag` = ∞;
`closest_row` = `null`;
`closest_idx` = `null`;
for r *in* `rows` **do**
    for i *in* $[0, floor(len(r)/2) - 1]$ **do**
        $j = \text{len}(r) - 1 - i$;
        Swap entry $\pi_r(i)$ and $\pi_r(j)$ of r in $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$;
        $c = D(\mathbb{P}, \mathbb{Q})$;
        if $|d - c| < |d - \texttt{closest\_mag}|$ **then**
            `closest_mag` = c;
            `closest_row` = r;
            `closest_idx` = i;
        end
        Swap entry $\pi_r(i)$ and $\pi_r(j)$ of r in $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$;
    end
end
i = `closest_idx`;
$j = \text{len(r)} - 1 - i$;
r = `closest_row`;
Swap entry $\pi_r(i)$ and $\pi_r(j)$ of r in $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$;
Output $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$;

where π_r is the index mapping that will sort the vector r. This search space is computationally feasible to search through as it only grows linearly with respect to the length of r with a complexity of $\mathcal{O}(|r|)$. Furthermore, the search space contains both r when it is "maximally" drifted and when it has the same permutation as in $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$, which makes it a suitable reduction of the original search space.

In order to find the permutations for each CPT row modified in Algorithm 1 that will lead to a DM $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$ with a divergence from $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$ close to d^*, Algorithm 2 first iterates through all the possible swaps it can make in each relevant CPT row, and only performs the swap if it results in $D(\mathbb{P}, \mathbb{Q}) \geq d^*$. This step attempts to reduce $D(\mathbb{P}, \mathbb{Q})$ to a point where any other possible swaps that can be performed will result in $D(\mathbb{P}, \mathbb{Q}) \leq d^*$. After that, Algorithm 2 iterates through all the possible swaps it can make again to find the swap that will result in the closest $D(\mathbb{P}, \mathbb{Q})$ to d^*. Algorithm 2 ends by performing that swap and returning the resulting DM $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$.

6.3 Generating Random Decomposable Models with a Limit on Treewidth

So far in this chapter, we assumed we already have a DM that we wish to modify. However, this DM needs to come from somewhere. One way to obtain this initial DM is to learn it from some existing dataset. The problem of learning a DM from data is a widely studied problem and there are already multiple existing methods to do so, such as Chordalysis (Petitjean et al. 2013; Petitjean et al. 2014; Petitjean & Webb, 2015; Petitjean et al. 2018; Webb & Petitjean, 2016).

Another way to obtain this initial DM is to synthetically generate it from scratch. Generating a random DM involves 2 steps:

1. first, generating a random chordal graph, and then
2. generating the parameters of the DM.

In Subsection 6.3.1 we summarise the existing literature on generating random chordal graphs and show how we can modify existing methods to generate random chordal graphs of a certain treewidth. Then in Subsection 6.3.2 we briefly discuss the method we chose to generate the parameters of a DM.

6.3.1 Generating Random Chordal Graphs

The problem of generating random chordal graphs can be re-framed as a problem of generating subtrees of a tree and constructing the intersection graph between

these subtrees (Şeker et al. 2017; Şeker et al. 2018). This stems from the fact that every chordal graph is exactly the intersection graph of a family of subtrees in an undirected tree (Gavril, 1974). Here, an intersection graph refers to a graph where each node represents a subtree, and an edge between 2 nodes indicates that the 2 corresponding subtrees intersect in the tree.

Therefore, we can generate any random tree and subtrees on it without much care and still obtain an intersection graph that is also a chordal graph. Specifically, in order to generate a chordal graph with n vertices, Şeker et al. (2017) proposed a general method with the following steps

1. generate a random tree T of n nodes uniformly,
2. create n random subtrees of T, $(T_1, \dots, T_n)$, of average size k,
3. output the intersection graph of the trees $(T_1, \dots, T_n)$ as the chordal graph $\mathcal{G}$.

Here n and k are parameters where n is the number of vertices in the final chordal graph $\mathcal{G}$, and k is a rough representation of the "edge density" of the $\mathcal{G}$.

Since, the only non-trivial step is the generation of the random subtrees, much of the literature has been focused on developing algorithms for generating subtrees that will result in chordal graphs that are "diverse" (Şeker et al. 2017; Şeker et al. 2018). One such algorithm that is the `GrowingSubtree` algorithm (Şeker et al. 2018) as shown in Algorithm 3.

Algorithm 3: GrowingSubtree

Data: A tree T of n nodes and a positive integer $k \leq n$
Result: A tuple of n non-empty subtrees of T of average size $\frac{k+1}{2}$
for $i = 1$ *to* n **do**
  Select a random node x of T and set $T_i = \{x\}$;
  Select a random integer k_i between 1 and k;
  for $j = 1$ *to* $k_i - 1$ **do**
    Select a random node y of T_i that has neighbours in T outside of T_i;
    Select a random neighbour z of y outside of T_i and add z to T_i;
  end
end
Output $(T_1, T_2, \dots, T_n)$;

We can limit the treewidth of the resulting intersection graph of $(T_1, \dots, T_n)$ by maintaining a count over each vertex in T representing the number of subtrees that vertex has appeared in so far. Once the number of subtrees that a vertex belongs to exceeds the target treewidth, we then disallow that vertex in T from being added to any new subtrees. See Algorithm 4 for this modified algorithm.

Algorithm 4: GrowingSubtreeTW

Data: A tree T of n nodes, a positive integer $k \leq n$, and the maximum treewidth ω

Result: A tuple of n non-empty subtrees of T of average size $\frac{k+1}{2}$ whose intersection graph has a treewidth no more than ω

$A \leftarrow$ copy of vertices in $T, V(T)$;
count $\leftarrow \{a => 0 \ for \ a \in A\}$;
for $i = 1$ *to* n **do**
    Select a random node x in A and set $T_i = \{x\}$;
    Select a random integer k_i between 1 and k;
    for $j = 1$ *to* $k_i - 1$ **do**
        Select a random node y of T_i that has neighbours in T outside of T_i that are also in A;
        Select a random neighbour z of y outside of T_i that is also in A and add z to T_i;
    end
    for $v \in V(T_i)$ **do**
        $count[v] += 1$;
        if $count[v] \geq \omega$ **then**
            pop(A, v);
        end
    end
end
Output $(T_1, T_2, \ldots, T_n)$;

We now report existing theorems that will help prove the correctness of Algorithm 4.

Theorem 15 (Helly's theorem (Helly, 1923; Horn, 1972)) A collection of closed connected segments on a line E has a nonempty intersection if and only if every pair of segments has a nonempty intersection.

Theorem 16 (Helly's Theorem for trees (Horn, 1972)) A collection of subtrees of a tree has at least one common node if and only if every pair of subtrees has at least one common node.

Corollary 2 (Treewidth limit to chordal graph generation) Algorithm 4 limits treewidth of resulting chordal graph to ω.

Proof Assume we have a tree T and ω subtrees $(T_1, \dots, T_\omega)$ on T that intersect each other. Since, $(T_1, \dots, T_\omega)$ all intersect each other, then by Theorem 16 there exist a non-empty set of vertices A that are common in each ω subtrees.

Assume we add the new subtree $T_{\omega+1}$ such that it intersects with all existing ω subtrees. Then by Theorem 16, there exist a non-empty set of vertices B that are common in each ω subtrees and also in subtree $T_{\omega+1}$.

However, any common vertex between $(T_1, \dots, T_\omega, T_{\omega+1})$ must also be a common vertex between $(T_1, \dots, T_\omega)$. Therefore, $B \subseteq A$ which implies that when adding subtree $T_{\omega+1}$, the subtree must contain vertices that are already in ω subtrees.

However, Algorithm 4 prohibits any new subtree to contain vertices that have already been in ω subtrees. Therefore, adding a subtree $T_{\omega+1}$ that intersects with all ω subtrees is not possible.

Consequently, if each subtree is a node in the final intersection graph, Algorithm 4 will prevent cliques of size greater than ω from forming. Therefore, the resulting chordal graph will have a maximum treewidth of ω.

Therefore, we now have an algorithm for generating random chordal graphs with a treewidth below some given target.

6.3.2 Generating Random Parameters for Decomposable Models

Given a chordal graph structure $\mathcal{G}$, we can create a new decomposable model $\mathbb{P}_\mathcal{G}$ by initializing the parameters of this model. Recall that a chordal graph $\mathcal{G}$ has a direct

$\mathbb{P}(\boldsymbol{X}_{\mathcal{C}-\text{pa}(\mathcal{C})}\|\boldsymbol{X}_{\text{pa}(\mathcal{C})}\|)$	$\boldsymbol{x}_{\text{pa}(\mathcal{C})}^{(1)}$	...	$\boldsymbol{x}_{\text{pa}(\mathcal{C})}^{(\|\text{pa}(\mathcal{C})\|)}$
$\boldsymbol{x}_{\mathcal{C}-\text{pa}(\mathcal{C})}^{(1)}$	$p_{1,1}$	$\cdots$	$p_{1,\|\text{pa}(\mathcal{C})\|}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\boldsymbol{x}_{\mathcal{C}-\text{pa}(\mathcal{C})}^{(\|\mathcal{C}-\text{pa}(\mathcal{C})\|)}$	$p_{\|\mathcal{C}-\text{pa}(\mathcal{C})\|,1}$	$\cdots$	$p_{\|\mathcal{C}-\text{pa}(\mathcal{C})\|,\|\text{pa}(\mathcal{C})\|}$

TABLE 6.2: Example of conditional probability table in a clique tree/forest.

clique tree/forest $\mathcal{T}$ representation. By choosing a random maximal clique in each tree of the clique forest to be the root clique, we can represent our parameters as a bunch of CPTs.

With this representation, we can initialise the parameters of a decomposable model by generating, for each clique $\mathcal{C}$ in the clique forest, vectors on the $|\mathcal{C} - \text{pa}(\mathcal{C})| - 1$ simplex for each possible value of the parent variables $\boldsymbol{X}_{\text{pa}(\mathcal{C})}$. The benefit of initialising the parameters in terms of CPTs is that it ensures consistency between the parameters of the model despite the columns of each CPT being initialised independently.

All we need now is a method to randomly generate a vector of length k that lies on the $k-1$ simplex. An easy way to do this is to just sample from a Dirichlet distribution with k concentration parameters, $\alpha_1, \ldots, \alpha_k$. These concentration parameters will then act as user-given parameters to our decomposable model generator.

6.4 Conclusion

To summarise, in this chapter we proposed a method to generate datasets containing occurrences of concept drift with known magnitudes. We achieved this by first proposing a method for modifying a DM such that the modified DM is a target amount of divergence away from the original DM. In other words, this proposed method induces a target amount of drift in a DM. Once we have modified the original DM to obtain a modified DMs, we can then construct a dataset using samples from both the original and modified DM. The occurrence of concept drift in this dataset is then the transition point between samples from the original DM to the modified DM, with the magnitude of this occurrence of concept drift being the divergence between the original and modified DM.

However, before tackling the problem of inducing some target amount of drift in a DM, we first made the argument that, when creating the underlying distributions for some dataset meant for testing methods to adapt to concept drift, it is important to control the entropy of these underlying distributions. Specifically, we argued that the entropy of a distribution can be used as measure of the "difficulty" in

modelling this distribution. The "difficulty" of a distribution is important when creating datasets meant for testing drift adaptation methods as, distributions that are too "easy", can cause all the adaptation techniques being tested to adapt too quickly, causing difficulties in differentiating between the performance of these adaptation techniques.

We proposed that one simple way to control for the entropy of the underlying distributions used to generate the test dataset, is to just ensure that each of the underlying distributions have the same entropy. In fact, we showed that, for any 1-dimensional discrete distribution, all permutations will have the same entropy. We then demonstrated how this fact can be applied to induce drift in the joint and conditional distributions of a DM without changing its entropy. This application basically involves permuting the rows of a specific set of CPTs encoded in the given DM. Specifically, this set of CPTs is obtained by taking the clique probability of each maximal clique in the DM and dividing them the separator probability of any minimal separators connected to the maximal clique.

Another aspect of inducing drift in DMs we had to consider, is the possibility that the target amount of drift we wish to induce might be higher than the maximum possible amount of drift we can actually induce. Therefore, we needed a method to find the maximum amount of drift we can induce in a given DM. To this end, we showed that, for any 1-dimensional discrete distribution, the permutation with the greatest $\alpha\beta$-divergence from the original permutation is the original permutation but in reverse sorted order. We can then apply this fact to decomposable models by applying this reverse sorted order on each row in the CPTs of the DM.

With these considerations and methods in mind, we then proposed a two-step approach to inducing drift in a DM. This approach involves first quickly achieving an amount of drift that is in the ballpark, but slightly above, the target drift magnitude. Algorithm 1 achieves this step by maximally drifting rows in the CPTs of the given DM by permuting these rows by reverse sorted order. If the maximum amount of drift we can induce in the DM is lower than the target drift magnitude, Algorithm 1 just returns a DM where all the rows of its CPT are drifted maximally, and the drift induction procedure ends. Otherwise, once the induced drift in the DM is slightly above the target drift magnitude, the second step involves finding permutations of the CPT rows that we modified in the first step. This step is difficult as the search space for each CPT row grows factorially with respect to the length of the CPT row. Therefore, Algorithm 2 carries out this step by only considering a subset of this factorial search space. This specific subset of the entire search space only grows linearly with respect with the CPT row length, and is therefore tractable.

Chapter 7

Application to Estimating Joint Divergences between Sample Data

As we established in Section 1.2, estimating the divergence between 2 distributions from sample data, hereby referred to as the *divergence estimation* problem, is an important problem in the field of machine learning. Due to its importance, there are numerous methods that have been developed to tackle this problem in the high-dimensional setting for discrete data. Most existing methods are general and do not make any assumptions regarding the variables and domains of these distributions. Instead they interpret the data from any discrete distribution as being from a uni-variate distribution with a support size S based on the number of unique items in the sampled data. However, in the worst case, the support size of any k-variate discrete distribution grows exponentially with respect to k. Furthermore, as we will show in Section 7.4, these existing methods do not scale well past 30 binary variables. The question then is, assuming these distributions have a large support due to having a large number of variables, k, is it possible to exploit this assumption to assist in the divergence estimation problem.

Therefore, given this assumption on the distributions, in this chapter, we will use DMs to essentially decompose the problem of divergence estimation into smaller sub-problems. Specifically, as we will further discuss in Section 7.2, our proposed approach will involve first learning 2 decomposable models using samples from each target distribution, and then computing the divergence between these models. An illustration of this general approach can be found in Figure 7.1.

7.1 Previous Work in Divergence Estimation

Estimating the divergence between 2 high-dimensional discrete distributions, $\mathbb{P}$ and $\mathbb{Q}$, with limited samples is a problem that has been addressed by a number of previous approaches (Acharya, 2018; Pavlichin et al. 2019; Z. Zhang & Grabchak, 2014; Han et al. 2016; Han et al. 2020; Bu et al. 2018; Jiao et al. 2018). In this section

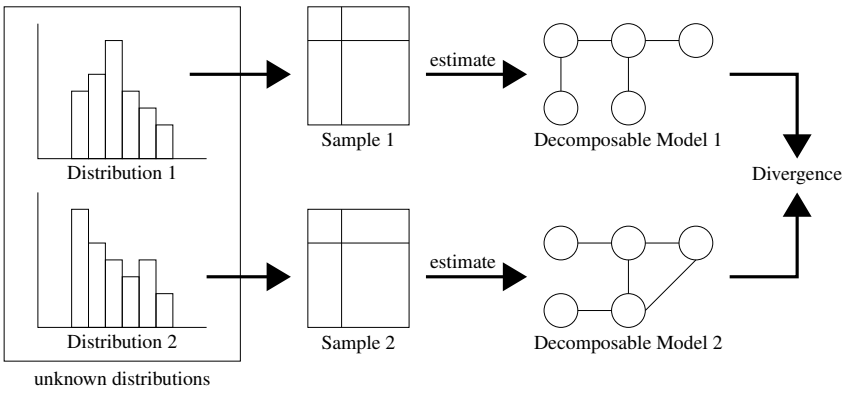


FIGURE 7.1: Overview of proposed approach for divergence estimation. The approach first learns 2 decomposable models from samples of 2 distributions. The divergence between the distributions is estimated by computing the divergence between the decomposable models.

we will discuss some of these previous approach to divergence estimation and how these approaches relate to one another.

A naive way to approach divergence estimation would be to use a plug-in approach where the population distributions are estimated using the empirical distribution, and these estimates are plugged in to the divergence we wish to estimate. However, instead of using the empirical distribution, it might be better to use a more sample-efficient method for estimating the distributions $\mathbb{P}$ and $\mathbb{Q}$ in high-dimensional scenarios. One such way is to use the Profile Maximum Likelihood (PML) to estimate the distributions $\mathbb{P}$ and $\mathbb{Q}$. The PML estimator maximises likelihood of observing the number of unique symbols appearing the amount of times within a dataset and better explains the data than standard Maximum Likelihood when the number of symbols is large compared to the size of the given dataset (Orlitsky et al. 2004). It was later shown that a PML-based approach is competitive for estimating any symmetric property of a collection of distributions, such as the $\alpha\beta$-divergence (Acharya, 2018). However, since the PML is difficult to compute exactly, approximations of the PML have been proposed that have been shown empirically to be competitive to existing divergence estimation approaches (Pavlichin et al. 2019).

Another approach to divergence estimation is to create an estimator of the target divergence. One work has shown that using a plug-in approach with the empirical distribution yields infinite bias error while the estimator they proposed has an exponentially decaying bias (Z. Zhang & Grabchak, 2014). More recently, a prominent approach to creating divergence estimators is the use of polynomial approximations of the divergence. One general approach that has been proposed is to identify "smooth" and "non-smooth" regimes of the divergence. This approach then uses a version of a bias-corrected Maximum Likelihood Estimator for estimation in

the "smooth" regime while using a polynomial approximation in the "non-smooth" regime (Han et al. 2016). This general approach can be used for estimating the Kullback-Leibler divergence, Hellinger distance, and χ^2-divergence (Han et al. 2016). A similar approach was proposed for estimating the Kullback-Leibler divergence with the main difference being the use of an bias-corrected augmented plug-in estimator in the "smooth" regime (Bu et al. 2018). This approach was also used for tackling the problem of estimation the Total Variation distance (L_1 distance) with the main difference being the use of a 2-dimensional polynomial approximation to approximate the L_1 distance in the "non-smooth" regime instead of just the best 1-dimensional polynomial approximation (Jiao et al. 2018).

One issue with the polynomial approximation approaches discussed is that they require splitting the available sample into 2 or more parts, the first for regime classification and the rest for estimation. Therefore, these polynomial approximation approaches don't fully use all the samples for estimation. Recently, a method was proposed that does not require splitting the sample in order to carry out regime classification. This method solves a problem-independent linear program based on moment matching to carry out estimation in the "non-smooth" regime while using a problem-dependent bias-corrected plug-in estimator in the "smooth" regime (Han et al. 2020).

7.2 Divergence Estimation using Decomposable Model: Decomposable Model Divergence Estimator (DMDE)

Recall from the beginning of this chapter that the divergence estimation problem involves the estimation of some divergence between distributions $\mathbb{P}$ and $\mathbb{Q}$ using only samples from $\mathbb{P}$ and $\mathbb{Q}$. Specifically in this chapter, we will tackle the problem of estimating the $\alpha\beta$-divergence, as defined in Definition 11, between $\mathbb{P}$ and $\mathbb{Q}$. We will apply a plug-in approach to this problem. This approach will involve:

1. learning 2 DMs in conjunction with their shared computation graph, $\mathcal{H}$, from the samples of $\mathbb{P}$ and $\mathbb{Q}$ respectively, and
2. computing the $\alpha\beta$-divergence between the joint distribution of these two DMs.

We will refer to this specific approach as the decomposable model divergence estimator (DMDE).

It is clear that the second step can be carried out by the method described in Chapter 3. However, the best means for achieving the first step is still unclear.

Specifically, it is still unclear how we might obtain the computation graph $\mathcal{H}$, in conjunction with learning the decomposable models from samples of $\mathbb{P}$ and $\mathbb{Q}$.

In this chapter we will use a straightforward approach to obtain the computation graph in conjunction with the decomposable models estimating distributions $\mathbb{P}$ and $\mathbb{Q}$. Specifically, we will learn the computation graph first by pooling the samples from both distributions together and using Chordalysis-SMT (Petitjean et al. 2013; Petitjean & Webb, 2015; Webb & Petitjean, 2016) to learn a chordal graph structure on this pooled dataset. We then use this chordal graph learnt on the pooled dataset as the computation graph $\mathcal{H}$. Furthermore, we will also use this chordal graph as the graphical structure of the decomposable models estimating the distributions $\mathbb{P}$ and $\mathbb{Q}$. After fixing the structure of these 2 decomposable models, we then learn the parameters of these decomposable models on samples from distributions $\mathbb{P}$ and $\mathbb{Q}$ respectively.

However, this approach may result in decomposable models with inaccurate structures because the variable structure can change due to concept drift. Ideally the decomposable models for the 2 distributions would be allowed to differ in structure. The methods we have developed can accommodate such differences in structure, so long as there is an appropriate method for finding a computation graph for the 2 decomposable models. We leave the problem of creating a method to achieve this to future research.

7.3 Datasets for Experiments

In order to run the experiments throughout the rest of the chapter, specifically in Sections 7.4 and 7.5, we require datasets that have data sampled from distributions with a known amount of divergence between them. Due to the scarcity of such high-dimensional datasets, we decided to use a synthetic dataset generator, such that each generated dataset pair (the datasets before and after drift) will contain data sampled from 2 distributions with a known divergence between them.

Creating such a generator, especially one that can generate high-dimensional datasets, is a problem in and of itself. One way to approach this problem is to create 2 generative models representing the 2 distributions the dataset pair will be sampled from. These generative models should be created such that the divergence between the 2 distributions matches some given target. Therefore, in order for a generator using this approach to be viable, we need to use a class of generative models with a tractable method for computing some divergence between 2 models under this class.

The problem lies in the fact that it is unclear whether there exists a method that is tractable in high dimensions, for computing the divergence between two models

under any class of generative models. That said, in this thesis, we have already developed a method for computing divergences between high-dimensional DMs in Chapter 3. Using this method for divergence computation, we have also developed a generator in Chapter 6 that can generate datasets with known magnitudes of concept drift. However, the generator in Chapter 6 uses DMs to represent and manipulate the underlying distribution, the same class of models used by our divergence estimation method DMDE. Ideally, we would use another class of generative models for this purpose in order to increase the fairness of the comparison experiments in Section 7.4. However, developing a method to measure a divergence under another class of generative models is out of scope of this thesis. Therefore, in order to slightly compensate for this source of unfairness, we will limit the treewidth of the decomposable models used by DMDE. By limiting the treewidth of the decomposable models learnt from the sampled data, we can observe how DMDE behaves when the learnt decomposable models cannot accurately model the original population distributions.

For the experiments in this the rest of this chapter, we will use the drift generator in Chapter 6, which we will refer to as decomposable model drift generator (DMDG), to create 50 dataset pairs for each combination of the parameters below. To be clear, the generator measures the true divergence between the 2 generating models while the estimators estimates that divergence from data generated from those models.

Hellinger distance 0.1, 0.3, 0.5, 0.7, 0.9

Number of Binary Variables 10, 15, 20, 30, 40, 50, 100

Treewidth 3, 5, 7, 9

Each dataset pair, representing data sampled from 2 distributions with a given Hellinger distance between them, will contain 100,000 samples from each distribution, totaling to 200,000 samples over both datasets in each pair.

We use the Hellinger distance as the target divergence as it is bounded between 0 and 1, as opposed to the Kullback-Leibler divergence which is unbounded. By having a fixed scale, we are able to ensure that the datasets we generate cover the entire range of possible Hellinger distances. This should allow for our experiments to have more coverage and therefore be more systematic. Furthermore, by having a fixed range of values, and therefore a somewhat universal meaning for any value of the Hellinger distance, we are able to compare estimation results across datasets of different configurations by focusing on specific values of Hellinger distances in the analysis of the results.

However, since we will compare DMDE against the other methods in Table 7.1 at estimating the Symmetric KL divergence, $KL(P\|Q) + KL(Q\|P)$, we will need

Method	Description	Reference
zg	estimator for KL divergence	Z. Zhang & Grabchak (2014)
bzlv	polynomial approximation	Bu et al. (2018)
hjw	polynomial approximation	Han et al. (2016)
jfc_tw=n	our proposed method but with a limit to the treewidth of the decomposable model of n	-

Table 7.1: Divergence estimation methods that have existing implementations that we will compare against. An implementation of our proposed approach and everything else needed to reproduce the experiments in this chapter can be found online at https://gitlab.com/lklee/div-est-via-discrete-decomp-models.

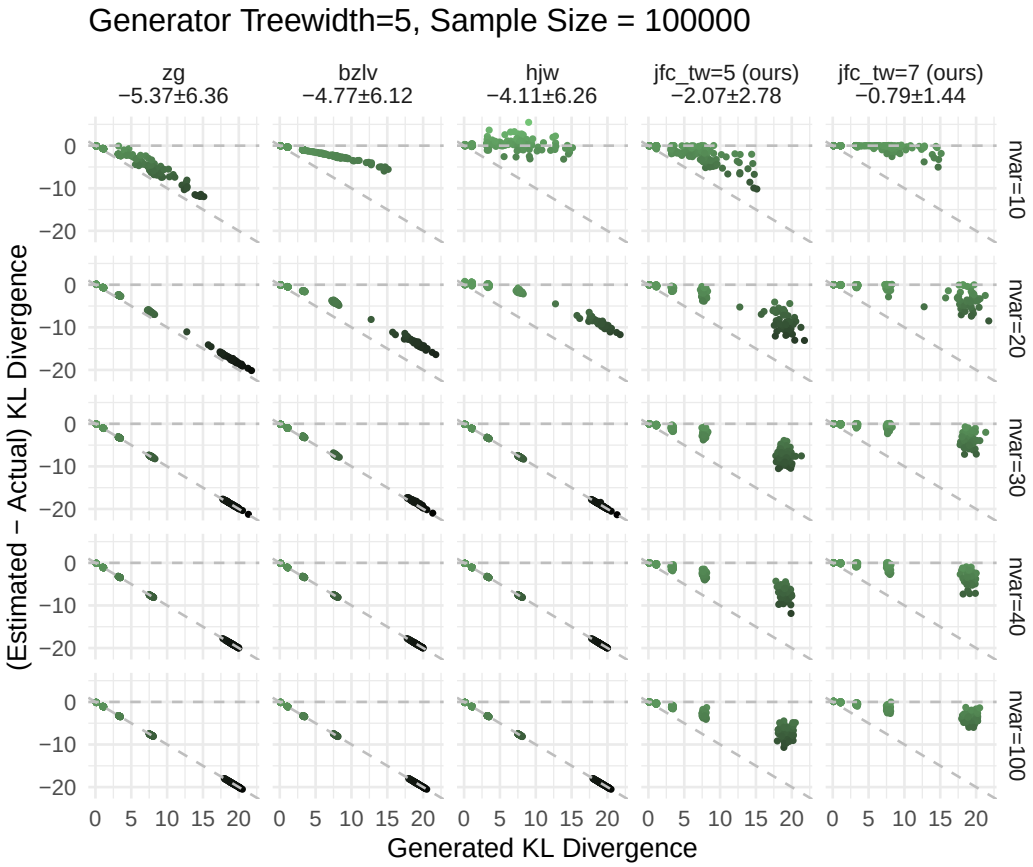
the true Symmetric KL divergence between the 2 population distributions of each dataset pair. Therefore, during the data generation process, we measure and record the exact Symmetric KL divergence between the decomposable models from which the dataset pairs are generated from. We also measure the exact Hellinger distance between the decomposable models during generation as well for use in Section 7.5.

7.4 Empirical Comparison with Previous Work

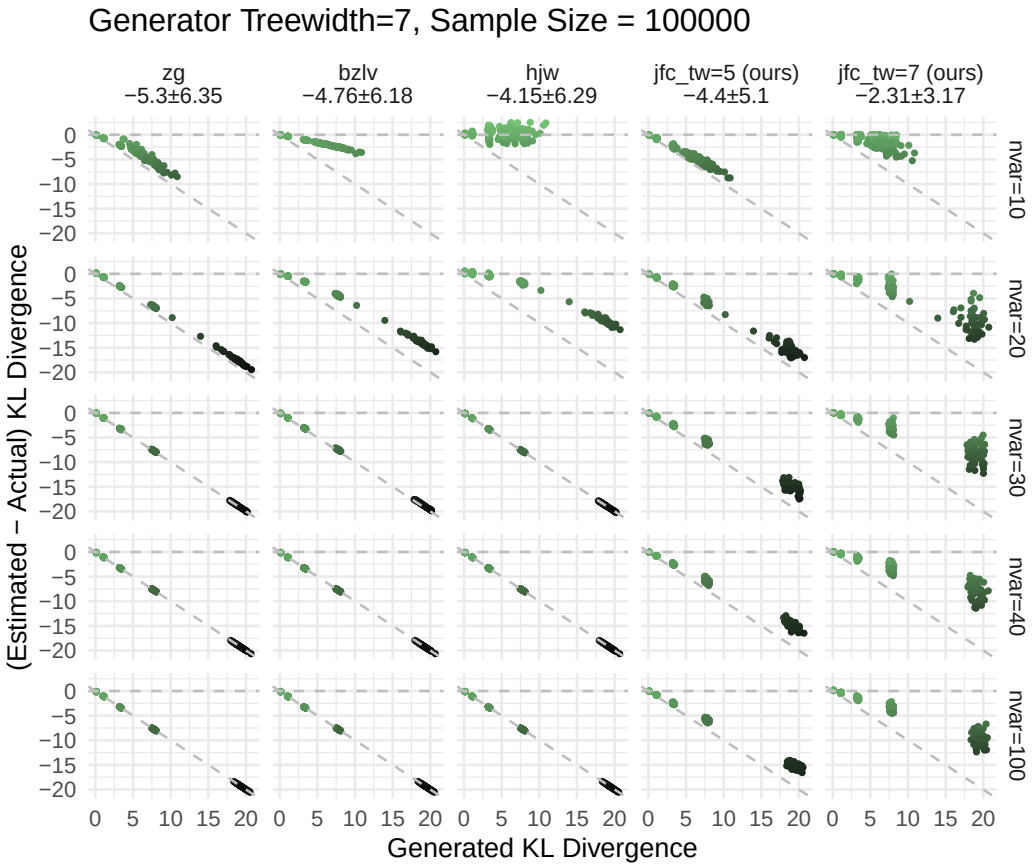
In this section we will compare the divergence estimator proposed in this chapter with various existing methods for estimating the divergence between 2 distributions using sample data. Specifically we will compare DMDE against the methods in Table 7.1, all of which have been described in Section 7.1, in estimating the symmetric Kullback-Leibler divergence between datasets in each dataset pair described in Section 7.3. We use the KL divergence in this comparison due to some of the methods in Table 7.1 only having implementations that estimate the KL divergence. We also use the symmetric version of the KL divergence to prevent any confusion on which of the 2 distributions is $\mathbb{P}$ and which is $\mathbb{Q}$.

It is important to reiterate that since both DMDG and DMDE uses DMs to represent probability distributions, it is possible for the results of this comparison experiment to be biased in favour of DMDE. Unfortunately, this is a problem that cannot be remedied in scenarios where the number of variables is large due to an absence of data generators capable of generating the required data. Therefore, the following results should be taken in with this fact in mind.

The results of the comparison between DMDE and the methods in Table 7.1 can be found in Figure 7.2. These results were obtained using all of the data available in each dataset, that is to say each method is given 100,000 samples from each of the 2



(A)



(B)

FIGURE 7.2
Comparison of DMDE with methods in Table 7.1. Each plot, comprised of multiple subplots, represents results from different treewidths (i.e. (A) 5 and (B) 7) for DMDG. The columns represent the methods being tested, and the rows, differing number of binary variables. The x-axis represents the symmetric KL-divergence generated, the y-axis represents the estimated minus true symmetric KL divergence. Points above and below zero on the y-axis represent over- and underestimation respectively. The line $y = -x$ represents max amount of underestimation error (when estimate given is 0).

distributions to estimate the symmetric KL-divergence between them. Furthermore, there are 2 plots in Figure 7.2, each representing results with DMDG of different treewidths, specifically treewidths of size 5 and 7.

The one thing we should take note of in Figure 7.2 is that distribution of points along the x-axis becomes increasingly concentrated at certain points as the number of variables increases. Specifically, when the number of variables is 100, there are 5 clusters along the x-axis which seem to correspond to the 5 levels of Hellinger distance used during the generation of the datasets in Section 7.3. It then seems probable that the high variability of the generated symmetric KL-divergence at low dimensions is due to inadequacies in generating datasets matching the 5 specified Hellinger distances at low dimensions. This inadequacy is likely due to a lack of parameters in the decomposable models to manipulate in order to achieve the target Hellinger distance.

On the surface, we can observe from Figure 7.2 that the method `zg` seems to severely underestimate the symmetric KL divergence between the distributions, giving estimates close to 0, even for datasets containing 10 binary variables. On the other hand, `bzlv` gives estimates that outperforms `zg` but still clearly underestimates the symmetric KL divergence at 10 binary variables. Out of all the existing methods in Table 7.1, `hjw` performs the best, and also does not greatly underestimate the KL divergence when the number of binary variables is 10. As the number of binary variables increases, these 3 existing methods start to increasingly underestimate the symmetric KL divergence. Specifically, these methods start returning estimates close of 0 when the number of binary variables is 30.

In terms of how DMDE performed, we can observe that the error for `jfc_tw=5` and `jfc_tw=7` does not increase with respect to the number of binary variables in the datasets. Instead, the error for both `jfc_tw=5` and `jfc_tw=7` increases as the treewidth of the data generator DMDG increases. We can also observe from `jfc_tw=7`'s results in Figure 7.2a that DMDE preforms very well when the treewidth limit of DMDE is slightly higher than the treewidth of the dataset's underlying distribution.

When the dimensionality is low (i.e. number of variables is 10 or 20), the existing methods, `zg`, `bzlv`, and `hjw`, give some of the most accurate estimates, regardless of the treewidth of DMDG. On the other hand, the results for our method, DMDE, on low dimensions is a bit of a mixed bag. In low dimensions, the existing methods are more accurate than DMDE whenever the treewidth is limited to be less than that of the generating distributions. However, even when that is not the case, DMDE performs slightly worse than `hjw` when the number of variables in the dataset is 10.

Where our method, DMDE, performs significantly better than the other methods is when dimensionality is high (i.e. where the number of variables is ≥ 30). As we

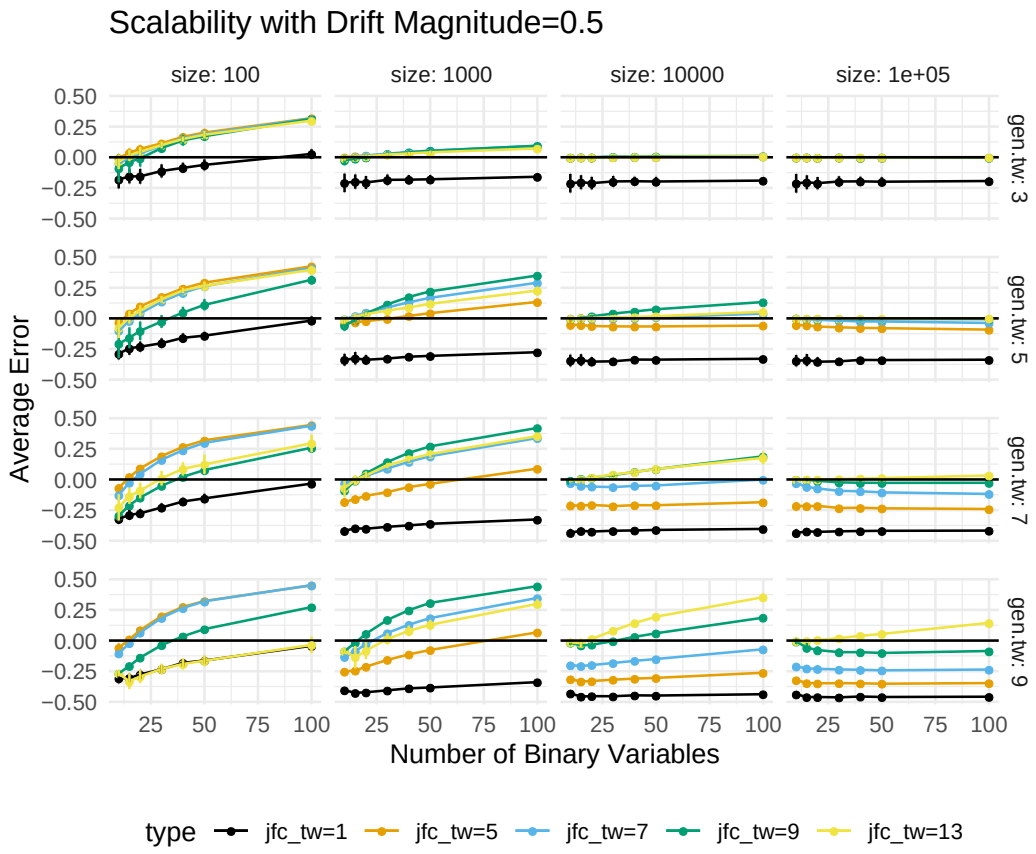
can observe from Figure 7.2, when dimensionality is high, `zg`, `hjw`, and `bzlv` start to return estimates that are close to 0. Conversely, a similar issue only occurs in DMDE when the treewidth limit given to DMDE is about 2 binary variables fewer than the treewidth of the source distribution. But even then, DMDE is still slightly more accurate than `zg`, `hjw`, and `bzlv`.

7.5 Characteristics of Our Method

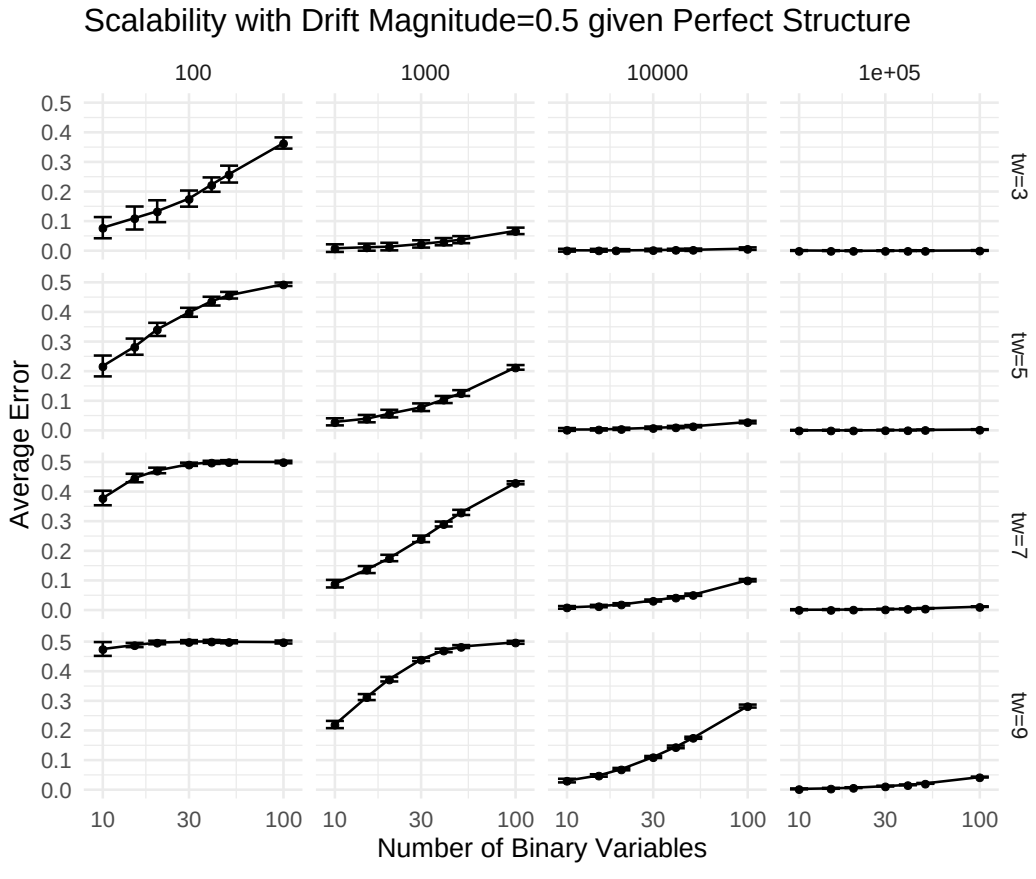
As stated at the start of this chapter, one of the main goals of our proposed method, DMDE, is to estimate $\alpha\beta$-divergences from sample data while scaling well with respect to the number of variables in the population distribution. Therefore, in this section we will specifically study how DMDE scales with respect to the dimensionality of the population distributions. Furthermore, we will use the estimation of the Hellinger distance from sample data as the target problem throughout this analysis so that we can compare estimation results between distributions of different configurations. For the rest of this section, we will only focus on the estimation from datasets with a Hellinger distance of 0.5. We believe this slice of the results is sufficient in conveying the relevant characteristics of DMDE's scalability with respect to the number of binary variables.

From Figure 7.3a, we can observe that in general the raw difference between the estimated and actual Hellinger distance increases monotonically as the number of binary variables increases. Furthermore, for DMDG treewidths of 3 and 5, the raw error is higher for DMDE with higher treewidths compared to DMDE with lower treewidths. However, for DMDG treewidths of 7 and 9, this is not necessarily the case. We can observe that for sample sizes of 100 and 1000, `jfc_tw=13` is not necessarily the method with the highest raw error, in fact for a sample size of 100, both `jfc_tw=5` and `jfc_tw=7` have higher raw errors than `jfc_tw=9` and `jfc_tw=13`.

However, observe that in Figure 7.3a, there's a general trend of DMDE actually increasing in accuracy as the number of variables increases. On the face of it, it is surprising that error should decrease for some configurations of DMDE as the number of variables increases, because it should be more difficult to model a high-dimensional distribution. Investigating this counter-intuitive result further, there are 2 things we can observe from Figure 7.3a. The first observation is that the initial estimate for most treewidth limits of DMDE underestimates the true Hellinger distance when the number of variables is small. The second observation is that the estimated Hellinger distance tends to increase as the number of variables increases, regardless of the initial estimate on low number of variables. Therefore, we hypothesise the following:



(A)



(B)

FIGURE 7.3
Scalability of DMDE (w.r.t number of binary variables) at estimating the Hellinger distance from sample data. Each row represents different DMDG treewidths, each column represents different sample sizes. The x-axes represent the number of binary variables, the y-axes represent the raw difference between the estimated and actual Hellinger distance. Each point is the mean over the 50 different datasets with error bars representing the standard deviation. (A) contains results when the DM is learnt from data while (B) contains results when the DMs used in DMDG are directly given to DMDE

Hypothesis 1 the final error we observe is comprised of 2 types of errors:

1. errors due to inaccuracies in learning the structure of the decomposable model (structure error)
2. errors due to inaccuracies in parameter estimation (parameter estimation error)

In order to verify Hypothesis 1 and understand how both these types of errors affect the divergence estimation error, we ran an experiment to study the effects of just the parameter estimation error on the divergence estimation error. This experiment is carried out by using the decomposable models created during the data generation process as the decomposable models of DMDE when estimating the Hellinger distance from sample data. This eliminates any need to learn the structure of the decomposable model from sample data and thereby prevents structure error from occurring.

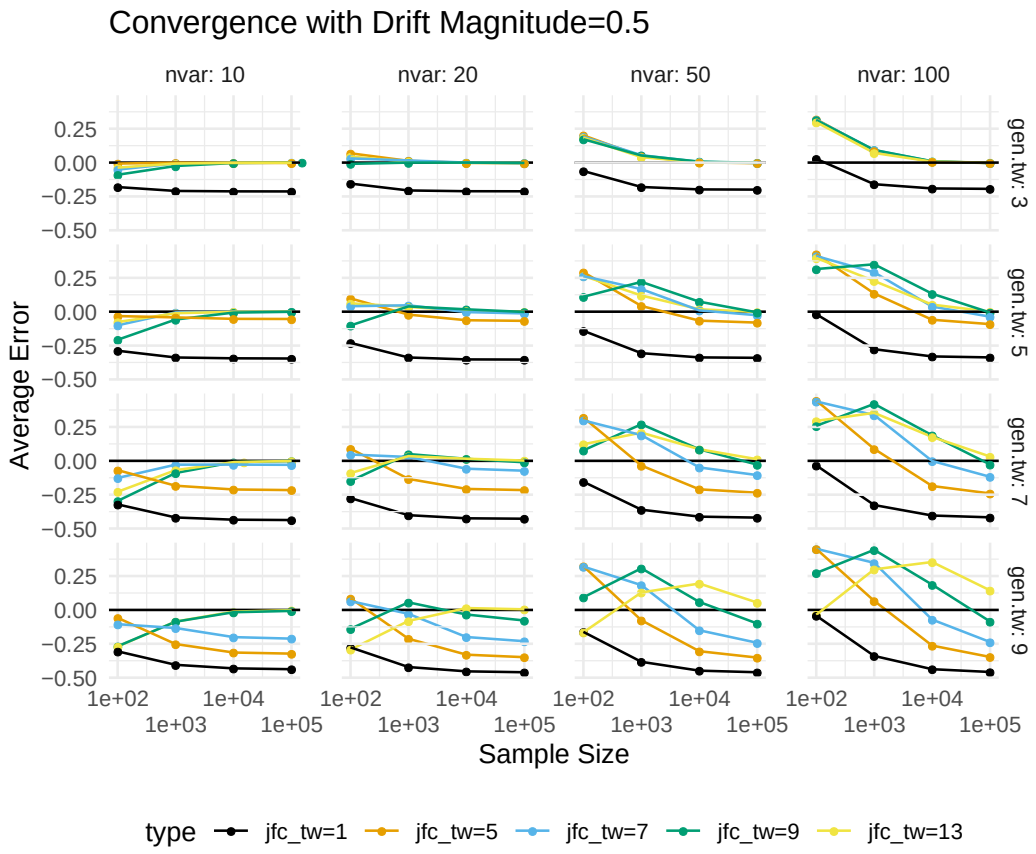
We can then use the results of this experiment to indirectly infer the effect structure error has on the divergence estimation error. We will not be studying the effect of structure error directly as it is hard to produce structure error without also producing parameter estimation error indirectly as well. Therefore, it is hard to devise experiments that study the effect of only structure error.

From Figure 7.3b, the first thing we observe is that the mean raw error decreases as we go from the leftmost column in the grid to the rightmost column. This is plainly just due to an increasing number of samples given to DMDE to estimate the Hellinger distance. Furthermore, we can also observe that as the number of variables in the source distribution increases, the mean error increases as well. This is likely due to parameter estimation errors over each maximal clique propagating and accumulating throughout the entire model. As the number of variables increases, so does the number of maximal cliques within the decomposable model. Therefore, even if the parameter estimation error within any maximal clique of the same size is similar, the total parameter estimation error across the entire model will increase.

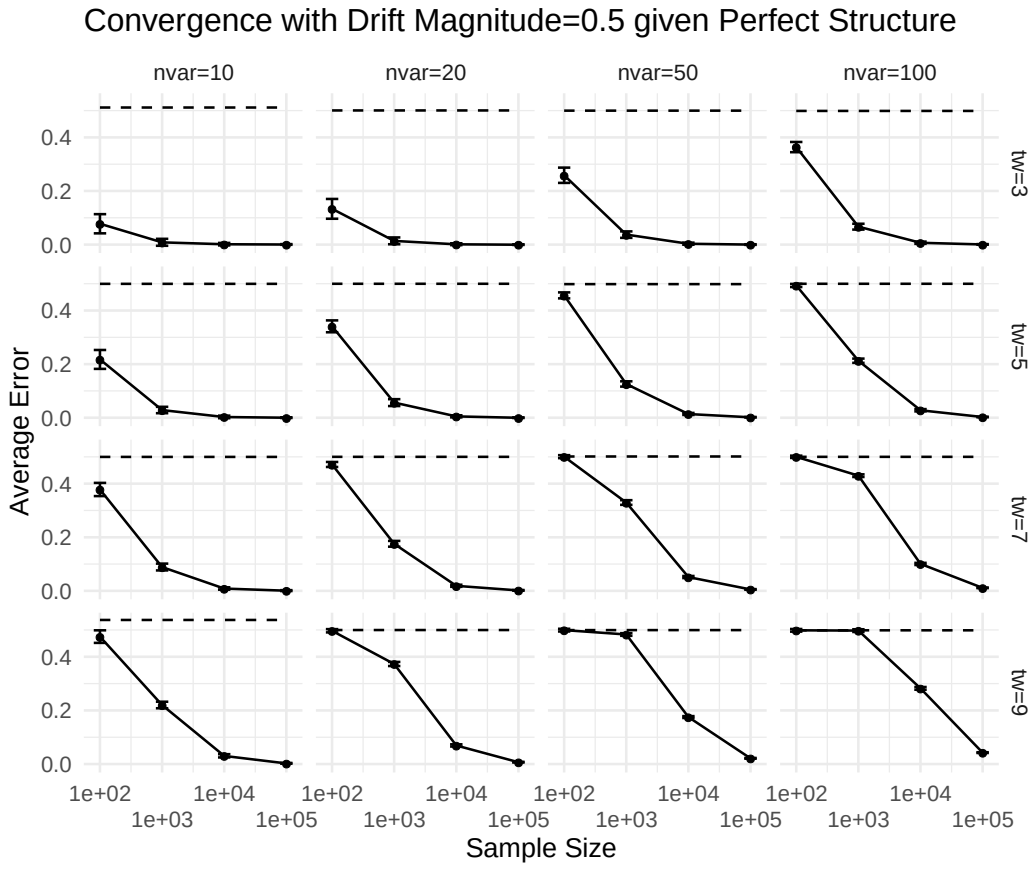
More importantly, we can observe from Figure 7.3b that if the structure of the decomposable models learnt by DMDE from data is perfect, DMDE consistently overestimates the true Hellinger distance in the dataset. From this observation, we propose the following hypothesis:

Hypothesis 2 Assume we have 2 distributions, p and q, represented by 2 Decomposable Models, $\mathcal{M}_P = (G_p, \pi_P)$ and $\mathcal{M}_Q = (G_q, \pi_q)$, and 2 datasets, D_p and D_q, sampled from p and q respectively.

Let $\hat{p}$ and $\hat{q}$ be the estimates of the distributions p and q respectively. Both $\hat{p}$ and



(A)



(B)

FIGURE 7.4
Convergence of DMDE at estimating Hellinger distance. Each row represents different generator treewidths, each column represents different numbers of binary variables. The y-axes represent the raw difference between the estimated and actual Hellinger distance, the x-axes represent different sample sizes taken from the start of the first and second half of the generated data. Each point is the mean over the 50 different datasets generated with error bars representing the standard deviation. (a) contains results when the decomposable model is learnt from data while (b) contains results when the decomposable models used in DMDG are directly given to DMDE

$\hat{q}$ are obtained using decomposable models, $\mathcal{M}_{\hat{p}} = (G_p, \hat{\pi}_p)$ and $\mathcal{M}_{\hat{q}} = (G_q, \hat{\pi}_q)$, learnt on the datasets D_p and D_q. Specifically, we fix the graph structure of both $\mathcal{M}_{\hat{p}}$ and $\mathcal{M}_{\hat{q}}$ to be the graph structure of the decomposable models representing the population distributions, G_p and G_q. The parameters, $\hat{\pi}_p$ and $\hat{\pi}_q$, are then learnt from the datasets D_p and D_q by learning the MLE of the multinomial distributions over each maximal clique and minimal separator of the graphs G_p and G_q.

Given this setup, we then hypothesise that the expected Hellinger distance between the estimates $\hat{p}$ and $\hat{q}$ over multiple realisations of D_p and D_q will be greater than the actual Hellinger distance between the true distributions p and q, or in other words:

$$\mathbb{E}_{D_p, D_q}[H(\hat{p}, \hat{q})] > H(p, q)$$

Unfortunately, we do not have a formal proof for Hypothesis 2, however given the observations from Figure 7.3a, it seems likely that this is true.

Assuming Hypothesis 2 is true, we then propose the following hypothesis to explain the underestimation occurring in Figure 7.3a:

Hypothesis 3 Errors caused by learning an incorrect structure for the decomposable models used by DMDE will, on average, cause DMDE to underestimate the true Hellinger distance, assuming sufficient data is provided to minimise parameter estimation error.

We can use both Hypothesis 2 and Hypothesis 3 to help explain some of the behaviour exhibited by DMDE in Figure 7.3a Specifically, from the first column of Figure 7.3a, representing results using just 100 data points from each population distribution, we can observe that `jfc_tw=13` increasingly underestimates the Hellinger distance as the treewidth given to DMDG increases. If Hypothesis 3 is true, then this increase in underestimation for `jfc_tw=13` should be due to increasing errors in the graph structure learnt by DMDE as DMDG's treewidth increases. Strangely enough, even though `jfc_tw=13` is the method with the least restriction on its treewidth, it seems to have the greatest error caused by learning an incorrect graph structure when the sample size is low and the DMDG treewidth is high. This seems to imply that learning an incorrect model can cause a greater amount of divergence estimation error than assuming a greater amount of independence between the variables being modelled.

Furthermore, we can observe a jump in overestimation for `jfc_tw=13` from sample size 100 to sample size 1000 when DMDG has a treewidth of 9. This will be easier to observe if we rearrange the axes of the plots in Figure 7.3 such that the x axis of

the plots represent the sample size used rather than the number of variables in the population distributions resulting in Figure 7.4a.

From Figure 7.4b we can see that, assuming DMDE has the same graph structure as the source distributions, DMDE will converge to the true Hellinger distance as the number of samples increases. This result should not be too surprising as when the sample size increases, the estimated parameters in the decomposable models approaches the true parameters, and therefore the decomposable models in DMDE approaches the population distributions.

With this fact in mind, the plot in Figure 7.4a can seem pretty counter-intuitive. Specifically, we can observe that for datasets with a high number of variables and a high treewidth for DMDG, the divergence estimation error of `jfc_tw=13` increases as the sample size increases and then later decreases back down again after a certain point. We can also observe the same phenomena occurring for `jfc_tw=9`. We hypothesise that the cause of this phenomena is due to the reduction of structure error as the sample size available to DMDE increases. Furthermore, we hypothesise that this reduction in structure error, which according to Hypothesis 3 causes underestimation, reduces at a greater rate than the parameter estimation error, which according to Hypothesis 2 causes overestimation. This gives the illusion that the divergence estimation error increases as the sample size increases, when really, we believe it is caused by the reduction of structure error, revealing the true error caused by parameter estimation.

7.6 Conclusion

In conclusion, we showed that it is possible to use DMs to leverage the inter-dependencies between variables in order to assist with estimating the divergence between 2 high-dimensional distributions using sample data. The advantage of our divergence estimation approach, DMDE, is that it scales with respect to the treewidth of the DMs learnt from the data. This is in contrast with a more direct approach, which has a complexity that scales exponentially with respect to the number of variables in the distributions of interest.

The work presented in this chapter represents a proof of concept that DMs can indeed assist in the problem of divergence estimation. Furthermore, despite only being a proof of concept, DMDE performs better than existing divergence estimation methods in datasets with more than 30 binary variables.

Through extensive experiments testing the behaviour of DMDE, we have also found that there are two main sources of errors in the divergence estimates that DMDE produces. The first source of error is due to inaccuracies in learning the independencies of the underlying distribution from samples data. Therefore, we call

this type of error "structure error". From our experiments, structure error seems to cause DMDE to underestimate the Hellinger distance between the underlying distributions of two samples. The second type of error, which we call "parameter estimation error", is error due to inaccuracies in estimating the parameters of the DM estimated from the given samples. We found that parameter estimation error will tend to cause DMDE to overestimate the Hellinger distance between the underlying distributions of two samples. The fact that these two types of errors, in estimating DMs from samples, can cause errors in estimating the Hellinger distance in opposing directions is interesting in and of itself, and therefore, more work is needed to further study the behaviour of divergence estimation techniques using a plug-in estimator approach with DMs.

Chapter 8

Closing Discussion and Conclusions

In summary, the main motivation of this thesis is to develop tools that might assist in tackling the problem of a distribution changing over some attribute such as location, time, population, and so on. This phenomena is known as concept drift, when changes occur over time, and concept shift, when changes occur over some other attribute. However, throughout this thesis, we used the term concept drift to refer to both types of distributional changes. There are many ways to characterise occurrences of concept drift in a dataset/stream. In this thesis, we mainly focused on the magnitude of these distributional changes, also known as drift magnitude. The magnitude of an occurrence of concept drift can be measured by computing the divergence between the underlying distributions before and after this occurrence of concept drift.

However, computing the divergence between two high-dimensional distributions directly is intractable. Therefore in this thesis, we tackled a relaxation of this problem, in which we assume that the underlying distributions we need to compute the divergence between can be modelled by chordal Markov networks (MNs). To this end we focused on developing methods to compute the $\alpha\beta$-divergence between two high-dimensional decomposable models (DMs). Specifically, we developed methods to compute the $\alpha\beta$-divergence between the joint, marginal, and conditional distributions encoded in the DMs of interest in an efficient manner. Developing these methods of divergence computation between DMs allowed us to not only measure the magnitude of drift within a dataset, but also go beyond simple measurement. Specifically, we showed that our divergence computation method can be used in generating datasets with occurrences of concept drift of known magnitudes.

8.1 Summary of Technical Contributions

One of the core contributions of this thesis is developing a method to compute the functional $\mathcal{F}$, in Definition 13, between discrete distributions represented by 2 high-dimensional DMs. The distributions that we can compute $\mathcal{F}$ between are

not limited to just the joint distribution of the DMs, but also the marginal and conditional distributions encoded within the DMs. Specifically, the method we have developed is equivalent to running the junction tree algorithm (JTA) on the computation graph $\mathcal{H}$ of both the DMs using factors that are specifically constructed using the clique probabilities encoded in the original DMs. Therefore, the actual contribution of our method are the steps needed to construct these factors and the proof that running the JTA using the factors on the computation graph $\mathcal{H}$ is equivalent to computing the functional $\mathcal{F}$ between distributions represented by these DMs.

There are many possible applications where the ability to compute the divergence between distributions of 2 DMs can be useful. In this thesis we explored how the methods developed in Chapters 3 to 5 can be applied to create a generator that generates data containing concept drift of known magnitudes. Specifically, we developed a method to modify a given decomposable model such that the modified model is a certain magnitude away from the original.

In the process of developing such a generator, we argue that ensuring the data before and after drift have the same entropy is important for creating datasets that are useful for testing methods that adapt to concept drift. Furthermore, we show that any permutation of a discrete distribution will have the same entropy and that the permutation with the greatest $\alpha\beta$-divergence relative to the original distribution is the distribution with the reverse sorted order of the original. We also propose a simple algorithm for finding permutations of the joint and conditional probabilities in a decomposable model such that the modified model is a certain drift magnitude away from the original.

For the purpose of providing our generator with random, synthetically created, DMs, we also propose an algorithm to generate random DMs. In the process, we extend existing algorithms for generating random chordal graphs in order to generate random chordal graphs with a treewidth below a given treewidth limit.

With the ability to compute the divergence between DMs, we then use this method in estimating the divergence between the underlying joint distributions from which the two samples have been generated, from the samples alone. This problem is also known as the divergence estimation problem. We use our method of divergence computation here by first learning DMs on the available samples and then computing the $\alpha\beta$-divergence between these DMs that we learned from the samples. We call this procedure of divergence estimation the decomposable model divergence estimator (DMDE). Our current version of DMDE is only an initial proof of concept, but we have shown that it already has the capability to outperform any other existing divergence estimation techniques on datasets with more than 30 binary variables. We also conduct a systematic study on the character of DMDE and found that its two source of errors comes from inaccuracies in estimating

either the graphical structure or parameters of the DM from data. Furthermore, we find that these two errors causes underestimation and overestimation, respectively, when estimating the Hellinger distance between the underlying distributions of two samples.

8.2 Code Implemented

Throughout this thesis, we have implemented some of our proposed methods and algorithms in code. Specifically, we implemented the method to compete the joint $\alpha\beta$-divergence between two DMs from Chapter 3 in Python. This implementation can be found in the repository https://gitlab.com/lklee/comp-div-dm.

Furthermore, we have also implemented the drift generator from Chapter 6. However, the code-base for this implementation is currently fragmented across multiple languages. The method for inducing drift in a DM has been implemented in Java while the random DM generator was implemented in Python and Julia instead. Therefore, we plan to consolidate the implementations of the drift generator from Chapter 6 into a single language, specifically Python.

Lastly, have we also implemented the divergence estimator DMDE from Chapter 7. This implementation is in Java and heavily builds upon the code base of Chordalysis (Petitjean et al. 2013; Petitjean & Webb, 2015; Webb & Petitjean, 2016) due to its reliance on needing a method to learn decomposable models from high-dimensional data efficiently. Therefore, the code-base for DMDE also contains yet another implementation of the methods in Chapter 3 for computing the joint divergence between DMs. The code-base for this implementation can be found at https://gitlab.com/lklee/div-est-via-discrete-decomp-models.

8.3 Future Work

There are multiple possible future avenues of work to further improve on the methods in this thesis. For instance, recall that core to our method, multi-graph aggregated sum-product (MGASP), for computing divergences between 2 DMs, $\mathbb{P}_{\mathcal{G}_P}$ and $\mathbb{Q}_{\mathcal{G}_Q}$, is finding the computation graph $\mathcal{H}$ between them. The straightforward way to obtain $\mathcal{H}$ is to triangulate the graph union of $\mathcal{G}_P$ and $\mathcal{G}_Q$. However, since finding the minimal triangulation of a graph is an **NP**-hard problem (Yannakakis, 1981; Heggernes, 2006), it is possible that the triangulation of $\mathcal{G}_P \cup \mathcal{G}_Q$ results in a computation graph with a large treewidth. This can present issues with regards to the tractability of MGASP as the JTA, and therefore MGASP, scales exponentially with respect to the treewidth of the computation graph. Therefore,

more work is needed to avoid situations where MGASP becomes intractable due to large treewidths. For instance, more sophisticated methods to obtain a computation graph between two chordal graphs can help in avoiding the creation of computation graphs that have a needlessly high treewidth. Another possible solution to this problem is to settle for an approximation of $\mathcal{F}$ between distributions of $\mathbb{P}_{\mathcal{G}_\mathbb{P}}$ and $\mathbb{Q}_{\mathcal{G}_\mathbb{Q}}$ by using approximate inference algorithms on the computation graph instead.

Furthermore, throughout the thesis, we only considered computing the divergence between discrete distributions. Therefore, more work is needed to investigate how one might extend this approach for divergence estimation to numeric or even mixed-type high-dimensional data. The current state of the art projects the data onto a lower-dimensional space, thereby potentially losing precision (Goldenberg & Webb, 2020). The ContCordalysis extension of Chordalysis for learning graphical models from numeric data may provide a path to achieving this (Rahman & Haffari, 2020).

With the ability to compute the divergence between DMs, we applied this method to developing DMDE, an estimator for the divergence between two distributions using only samples of these distributions. However, there are many possible avenues of further research that can be carried out to extend and improve upon the proposed divergence estimator DMDE. For instance, in Chapter 7, we used the empirical distribution to estimate the clique probabilities in the decomposable models learnt from the data. However, it is possible to use more statistically efficient estimators for clique probability estimation, such as the Profile Maximum Likelihood mentioned in Section 7.1, and more work is needed to find, adapt, and test existing estimators on this approach of divergence estimation. Furthermore, when obtaining the computation graph in conjunction with the decomposable models from the available samples, we assumed that variable structure of the 2 distributions are the same. However, this assumption does not necessarily hold in real data. Our methods are already capable of estimating divergences between distributions modelled by decomposable models with different structures, save for the missing link of a robust method for deriving useful computation graphs for such models. Therefore, more work is needed to develop more sophisticated techniques for obtaining the computation graph in conjunction with the decomposable models estimating the distributions of interest.

One benefit of using our method of divergence computation between DMs for divergence estimation, is the ability to quickly compute the marginal divergence over variables that are confined within a single maximal clique in both DMs. This ability might be useful in the problem of identifying variable subsets, or subspaces, within the underlying distribution that contribute the greatest to the overall divergence/drift. Tackling this problem is especially pertinent in high dimensions as it seems reasonable that concept drift might only occur in certain subspaces while other subspaces remain relatively stable. By having knowledge on which subspace

of the distribution is responsible for the overall drift, this method is able to assist in drift adaptation by showing what parts of the model needs to be retrained, and which parts can be kept the same.

When trying to adapt to distributional changes over time, it might be possible for some distribution, i.e. concept, to reoccur again some time in the future. This type of concept drift is also known as recurring concept drift (Gonçalves Jr & Barros, 2013; Anderson et al. 2019). Being able to recognise previous concepts can greatly assist drift adaptation methods in quickly adapting to the recurring concept. Therefore, by being able to compactly store representations of the different concepts/distributions as DMs, and to compute the divergence between these DMs, we are able to identify if the current distribution has occurred before, and if so, recover a representation of the distribution that has been estimated from a greater number of samples.

It is important to study how different drift adaptation techniques adapt to drift of different magnitudes as these adaptation techniques might react to differing amounts of drift magnitude in different and unexpected ways (Webb & Petitjean, 2016). Unfortunately, prior to this thesis, most studies into the behaviour of existing drift adaptation techniques have not controlled for the drift magnitude of the occurrence of concept drift. Therefore, using the drift generator from Chapter 6 we can create datasets that contain known magnitudes of concept drift for this purpose. Specifically, we can either use, as input to the drift generator, a synthetically generated DM, resulting in a totally synthetic dataset, or a DM learnt from some existing dataset, resulting in a semi-synthetic dataset that we can use to test drift adaptation techniques with. However, there might be cases where we want to test some drift adaptation techniques on a totally unmodified dataset. In such cases, we can first map out and visualise the occurrences of concept drift, and their magnitudes, within the existing dataset using divergence estimation techniques such as the divergence estimator we proposed in Chapter 7. We can then use this "drift map" as a guide for analysing the behaviour of the adaptation techniques being tested on the dataset.

In general, a motivation for this work was the belief that the ability to assess divergences in high-dimensional data will have many applications in the field of concept drift. As we have discussed, some of these applications include:

1. more precise and nuanced concept drift detection by finding subspaces in the distribution that contribute most to the overall concept drift, which will allow selective retraining of only certain parts of the model, thereby increasing retraining efficiency,

2. more responsive and targeted approaches to drift recovery by identifying reoccurring concepts, allowing for the reuse of previously trained models that have been trained on a greater amount of data, thereby shortening the time it takes to fully adapt to a reoccurring concept, and

3. the creation of datasets with known magnitudes of concept drift, either via synthetic or semi-synthetic generation, or by initially mapping out and measuring any occurrences of concept drift in some existing dataset, for the purposes of assessing the performance and behaviour of various drift adaptation techniques in the face of varying magnitudes of concept drift.

These applications only cover just some of the possible applications of the methods developed in this thesis in the field of concept drift, let alone applications in other fields of machine learning such as transfer learning.

Nomenclature

Graphs

$\boldsymbol{C}(\mathcal{G})$ Maximal cliques of chordal graph $\mathcal{G}$

$\mathcal{G}$ Undirected Graph

$\mathcal{N}_{\mathcal{G}}(v)$ The set of vertices neighbouring v in graph $\mathcal{G}$.

$\boldsymbol{S}(\mathcal{G})$ Minimal separators of chordal graph $\mathcal{G}$

$\mathcal{T}$ A graph with no cycles, i.e. a tree

$\boldsymbol{\mathcal{T}}$ A collection of trees, $\boldsymbol{\mathcal{T}} = \{\mathcal{T}_1, \mathcal{T}_2, ...\}$, i.e. a forest.

$E(\mathcal{G})$ Edges of graph $\mathcal{G}$

$V(\mathcal{G})$ Vertices of graph $\mathcal{G}$

Information Theory

$D(\mathbb{P}, \mathbb{Q})$ Divergence between distributions $\mathbb{P}$ and $\mathbb{Q}$

$D_{AB}^{(\alpha,\beta)}(\mathbb{P}, \mathbb{Q})$ $\alpha\beta$-divergence between distributions $\mathbb{P}$ and $\mathbb{Q}$

$H(\mathbb{D})$ Shannon entropy of discrete distribution $\mathbb{D}$

Probabilistic Model

$\mathbb{D}_{\mathcal{G}}$ Decomposable model with graphical structure $\mathcal{G}$ representing the distribution $\mathbb{D}$

Other symbols

$\mathcal{F}$ Functional between 2 discrete probability distributions as defined in Definition 13

$\mathcal{F}_{Y|Z}$ The conditional functional between 2 discrete conditional distributions over Y given Z as defined in Proposition 9

Probability & Data

$\mathcal{X}_A$ State space of random variables $\boldsymbol{X}_A$

$\mathcal{X}_v$ State space of random variable X_v

$\mathbb{P}, \mathbb{Q}, \mathbb{D}$ Probability distributions over the variables $\boldsymbol{X}$

$\boldsymbol{X}_A$ Random variables with labels in the set A

x_A An element from the set $\mathcal{X}_A$

X_v Discrete random variable with label v

x_v An element from the set $\mathcal{X}_v$

Glossary

$\mathcal{B}$-partition	A set containing sets in an $\mathcal{N}$-partition that has vertices associated with some variable in a given set of variables $\boldsymbol{Y}$. See Definition 23.
$\mathcal{N}$-graph	Graph obtained from using function Γ defined in Definition 20.
$\mathcal{N}$-partition	Set of sets of vertices in a chordal graph that partitions both the maximal clique of the graph and some given subset of the vertices of the graph. See Definition 18.
concept drift	Formally defined as a change in the conditional distribution of some target/class variables given the covariate variables. However, in this thesis, we use the term "concept drift" to mean both concept drift and shift unless specified otherwise.
drift magnitude	Magnitude of an occurrence of concept drift or shift. Typically measured via the divergence between the distribution before and after drift.
`bzlv`	Method to estimate KL-divergence from Bu et al. (2018).
`hjw`	Method to estimate KL-divergence from Han et al. (2016).
`mcgo`	Method to compute the Kullback-Leibler (KL) divergence between 2 Bayesian networks (BNs) of different structures in (Moral et al. 2021).
`pgmpy`	Python library for graphical models (Ankan & Panda, 2015).
`zg`	Method to estimate KL-divergence from Z. Zhang & Grabchak (2014).
drift generator	Data generators capable of creating datasets with known occurrence of concept drift.

inter-attribute structure	The relationships and structure between a set of variables.
junction tree/forest	Junction tree/forest of a chordal graph.
marginal $\mathcal{N}$-probability	Marginal probability over the variables of a vertex set from some $\mathcal{N}$-partition defined in Definition 19.

Acronyms

BN	Bayesian network
CPT	conditional probability table
DAG	directed acyclic graph
DG	directed graph
DM	decomposable model
DMDE	decomposable model divergence estimator
DMDG	decomposable model drift generator
JTA	junction tree algorithm
KL	Kullback-Leibler
MGASP	multi-graph aggregated sum-product
MN	Markov network
pmf	probability mass function
rv	random variable
UG	undirected graph

Bibliography

Abdullah, Amirali; Kumar, Ravi; McGregor, Andrew; Vassilvitskii, Sergei; Venkatasubramanian, Suresh (2016). "Sketching, Embedding and Dimensionality Reduction in Information Theoretic Spaces". In: *Artificial Intelligence and Statistics*. May 2, 2016, pp. 948–956. URL: http://proceedings.mlr.press/v51/abdullah16.html (visited on 02/02/2020).

Acharya, Jayadev (2018). "Profile Maximum Likelihood Is Optimal for Estimating KL Divergence". In: *2018 IEEE International Symposium on Information Theory (ISIT)*. June 2018, pp. 1400–1404. DOI: 10.1109/ISIT.2018.8437461.

Alippi, Cesare; Boracchi, Giacomo; Carrera, Diego; Roveri, Manuel (2016). "Change Detection in Multivariate Datastreams: Likelihood and Detectability Loss". In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. IJCAI'16. New York, New York, USA: AAAI Press, July 9, 2016, pp. 1368–1374. ISBN: 978-1-57735-770-4.

Amari, Shun-ichi (2016). *Information Geometry and Its Applications*. Springer, Feb. 2, 2016. 378 pp. ISBN: 978-4-431-55978-8. Google Books: UkSFCwAAQBAJ.

Anderson, Robert; Koh, Yun Sing; Dobbie, Gillian; Bifet, Albert (2019). "Recurring Concept Meta-Learning for Evolving Data Streams". In: *Expert Systems with Applications* 138 (Dec. 30, 2019), p. 112832. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2019.112832.

Ankan, Ankur; Panda, Abinash (2015). "Pgmpy: Probabilistic Graphical Models Using Python". In: *Proceedings of the 14th Python in Science Conference* (2015), pp. 6–11. DOI: 10.25080/Majora-7b98e3ed-001.

Baidari, Ishwar; Honnikoll, Nagaraj (2021). "Bhattacharyya Distance Based Concept Drift Detection Method for Evolving Data Stream". In: *Expert Systems with Applications* 183 (Nov. 30, 2021), p. 115303. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2021.115303.

Balzanella, Antonio; Rivoli, Lidia; Verde, Rosanna (2013). "Data Stream Summarization by Histograms Clustering". In: *Statistical Models for Data Analysis*. Ed. by Paolo Giudici; Salvatore Ingrassia; Maurizio Vichi. Studies in Classification, Data Analysis, and Knowledge Organization. Heidelberg: Springer International Publishing, 2013, pp. 27–35. ISBN: 978-3-319-00032-9. DOI: 10.1007/978-3-319-00032-9_4.

Barz, Björn; Rodner, Erik; Garcia, Yanira Guanche; Denzler, Joachim (2019). "Detecting Regions of Maximal Divergence for Spatio-Temporal Anomaly Detection". In:

IEEE Transactions on Pattern Analysis and Machine Intelligence 41.5 (May 2019), pp. 1088–1101. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2018.2823766.

Berry, Anne; Blair, Jean R. S.; Heggernes, Pinar; Peyton, Barry W. (2004). "Maximum Cardinality Search for Computing Minimal Triangulations of Graphs". In: *Algorithmica* 39.4 (Aug. 1, 2004), pp. 287–298. ISSN: 1432-0541. DOI: 10.1007/s00453-004-1084-3.

Bhattacharya, Arnab; Kar, Purushottam; Pal, Manjish (2009). "On Low Distortion Embeddings of Statistical Distance Measures into Low Dimensional Spaces". In: *Database and Expert Systems Applications*. Ed. by Sourav S. Bhowmick; Josef Küng; Roland Wagner. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2009, pp. 164–172. ISBN: 978-3-642-03573-9. DOI: 10.1007/978-3-642-03573-9_13.

Bhattacharyya, Rishiraj; Chakraborty, Sourav (2018). "Property Testing of Joint Distributions Using Conditional Samples". In: *ACM Transactions on Computation Theory* 10.4 (Aug. 22, 2018), 16:1–16:20. ISSN: 1942-3454. DOI: 10.1145/3241377.

Bifet, Albert; Holmes, Geoff; Pfahringer, Bernhard; Gavaldà, Ricard (2009a). "Improving Adaptive Bagging Methods for Evolving Data Streams". In: *Proceedings of the 1st Asian Conference on Machine Learning: Advances in Machine Learning*. ACML '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 23–37. ISBN: 978-3-642-05223-1. DOI: 10.1007/978-3-642-05224-8_4.

Bifet, Albert; Holmes, Geoff; Pfahringer, Bernhard; Kirkby, Richard; Gavaldà, Ricard (2009b). "New Ensemble Methods for Evolving Data Streams". In: ACM Press, 2009, p. 139. ISBN: 978-1-60558-495-9. DOI: 10.1145/1557019.1557041.

Blair, Jean R. S.; Peyton, Barry (1993). "An Introduction to Chordal Graphs and Clique Trees". In: *Graph Theory and Sparse Matrix Computation*. Ed. by Alan George; John R. Gilbert; Joseph W. H. Liu. The IMA Volumes in Mathematics and Its Applications. New York, NY: Springer, 1993, pp. 1–29. ISBN: 978-1-4613-8369-7. DOI: 10.1007/978-1-4613-8369-7_1.

Bleuler, Cédric; Lapidoth, Amos; Pfister, Christoph (2020). "Conditional Rényi Divergences and Horse Betting". In: *Entropy* 22.3 (3 Mar. 2020), p. 316. ISSN: 1099-4300. DOI: 10.3390/e22030316.

Bonnici, Vincenzo (2020). "Kullback-Leibler Divergence between Quantum Distributions, and Its Upper-Bound". Dec. 10, 2020. arXiv: 2008.05932 [quant-ph]. URL: http://arxiv.org/abs/2008.05932 (visited on 03/01/2022).

Boracchi, Giacomo; Carrera, Diego; Cervellera, Cristiano; Macciò, Danilo (2018). "QuantTree: Histograms for Change Detection in Multivariate Data Streams". In: *International Conference on Machine Learning*. July 3, 2018, pp. 639–648. URL: http://proceedings.mlr.press/v80/boracchi18a.html (visited on 02/03/2020).

Bu, Yuheng; Zou, Shaofeng; Liang, Yingbin; Veeravalli, Venugopal V. (2018). "Estimation of KL Divergence: Optimal Minimax Rate". In: *IEEE Transactions on Information Theory* 64.4 (Apr. 2018), pp. 2648–2674. ISSN: 1557-9654. DOI: 10.1109/TIT.2018.2805844.

Bulatov, Andrei; Grohe, Martin (2004). "The Complexity of Partition Functions". In: *Automata, Languages and Programming*. Vol. 3142. Lecture Notes in Computer Science. Heidelberg, Germany: Springer, 2004, pp. 294–306.

Cai, Changxiao; Verdú, Sergio (2019). "Conditional Rényi Divergence Saddlepoint and the Maximization of α-Mutual Information". In: *Entropy* 21.10 (10 Oct. 2019), p. 969. ISSN: 1099-4300. DOI: 10.3390/e21100969.

Carrera, Diego (2020). "Learning and Adaptation to Detect Changes and Anomalies in High-Dimensional Data". In: *Special Topics in Information Technology*. Ed. by Barbara Pernici. SpringerBriefs in Applied Sciences and Technology. Cham: Springer International Publishing, 2020, pp. 63–75. ISBN: 978-3-030-32094-2. DOI: 10.1007/978-3-030-32094-2_5.

Carrera, Diego; Boracchi, Giacomo (2018). "Generating High-Dimensional Datastreams for Change Detection". In: *Big Data Research*. Selected Papers from the 2nd INNS Conference on Big Data: Big Data & Neural Networks 11 (Mar. 1, 2018), pp. 11–21. ISSN: 2214-5796. DOI: 10.1016/j.bdr.2017.09.001.

Chen, Yukun; Ye, Jianbo; Li, Jia (2020). "Aggregated Wasserstein Distance and State Registration for Hidden Markov Models". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.9 (Sept. 2020), pp. 2133–2147. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2019.2908635.

Choi, Arthur; Chan, Hei; Darwiche, Adnan (2005). "On Bayesian Network Approximation by Edge Deletion". In: *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*. UAI'05. Arlington, Virginia, USA: AUAI Press, July 26, 2005, pp. 128–135. ISBN: 978-0-9749039-1-0.

Cichocki, Andrzej; Cruces, Sergio; Amari, Shun-ichi (2011). "Generalized Alpha-Beta Divergences and Their Application to Robust Nonnegative Matrix Factorization". In: *Entropy* 13.1 (1 Jan. 2011), pp. 134–170. DOI: 10.3390/e13010134.

Cover, Thomas M.; Thomas, Joy A. (2006). *Elements of Information Theory*. 2nd edition. Hoboken, N.J: Wiley-Interscience, July 18, 2006. 784 pp. ISBN: 978-0-471-24195-9.

Cowell, Robert G.; Lauritzen, Steffen L.; David, A. Philip; Spiegelhalter, David J. (1999). *Probabilistic Networks and Expert Systems*. Ed. by V. Nair; J. Lawless; M. Jordan. 1st. Berlin, Heidelberg: Springer-Verlag, 1999. ISBN: 0387987673.

Csiszar, I. (1995). "Generalized Cutoff Rates and Renyi's Information Measures". In: *IEEE Transactions on Information Theory* 41.1 (Jan. 1995), pp. 26–34. ISSN: 1557-9654. DOI: 10.1109/18.370121.

David Destephen Lavaire, Jorge; Singh, Anshuman; Yousef, Mahmoud; Singh, Sumi; Yue, Xiaodong (2015). "Dimensional Scalability of Supervised and Unsupervised Concept Drift Detection: An Empirical Study". In: *2015 IEEE International Conference on Big Data (Big Data)*. Oct. 2015, pp. 2212–2218. DOI: 10.1109/BigData.2015.7364009.

Dawid, A. Philip (1979). "Conditional Independence in Statistical Theory". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 41.1 (1979), pp. 1–31. ISSN: 0035-9246. JSTOR: 2984718.

Dawid, A. Philip (1980). "Conditional Independence for Statistical Operations". In: *The Annals of Statistics* 8.3 (1980), pp. 598–617. ISSN: 0090-5364. JSTOR: 2240595.

Dechter, Rina (2003). *Constraint processing*. Elsevier Morgan Kaufmann, 2003. ISBN: 978-1-55860-890-0. URL: http://www.elsevier.com/wps/find/bookdescription.agents/678024/description.

Ditzler, Gregory; Polikar, Robi (2011). "Hellinger Distance Based Drift Detection for Nonstationary Environments". In: *2011 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE)*. Apr. 2011, pp. 41–48. DOI: 10.1109/CIDUE.2011.5948491.

Fehr, Serge; Berens, Stefan (2014). "On the Conditional Rényi Entropy". In: *IEEE Transactions on Information Theory* 60.11 (Nov. 2014), pp. 6801–6810. ISSN: 1557-9654. DOI: 10.1109/TIT.2014.2357799.

Gama, João; Žliobaite, Indre; Bifet, Albert; Pechenizkiy, Mykola; Bouchachia, Abdelhamid (2014). "A Survey on Concept Drift Adaptation". In: *ACM Comput. Surv.* 46.4 (Mar. 2014), 44:1–44:37. ISSN: 0360-0300. DOI: 10.1145/2523813.

Gavril, Fănică (1974). "The Intersection Graphs of Subtrees in Trees Are Exactly the Chordal Graphs". In: *Journal of Combinatorial Theory, Series B* 16.1 (Feb. 1, 1974), pp. 47–56. ISSN: 0095-8956. DOI: 10.1016/0095-8956(74)90094-X.

Ghifary, Muhammad; Kleijn, W. Bastiaan; Zhang, Mengjie; Balduzzi, David (2015). "Domain Generalization for Object Recognition with Multi-task Autoencoders". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015 IEEE International Conference on Computer Vision (ICCV). Dec. 2015, pp. 2551–2559. DOI: 10.1109/ICCV.2015.293.

Goldenberg, Igor; Webb, Geoffrey I. (2019). "Survey of Distance Measures for Quantifying Concept Drift and Shift in Numeric Data". In: *Knowledge and Information Systems* 60.2 (Aug. 1, 2019), pp. 591–615. ISSN: 0219-3116. DOI: 10.1007/s10115-018-1257-z.

– (2020). "PCA-based Drift and Shift Quantification Framework for Multidimensional Data". In: *Knowledge and Information Systems* 62.7 (July 1, 2020), pp. 2835–2854. ISSN: 0219-3116. DOI: 10.1007/s10115-020-01438-3.

Gonçalves Jr, Paulo Mauricio; Barros, Roberto Souto Maior de (2013). "RCD: A Recurring Concept Drift Framework". In: *Pattern Recognition Letters* 34.9 (July 1, 2013), pp. 1018–1025. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2013.02.005.

Hammersley, John Michael; Clifford, Peter (1971). "Markov Fields on Finite Graphs and Lattices". In: *Unpublished manuscript* (1971). URL: https://www.semanticscholar.org/paper/Markov-fields-on-finite-graphs-and-lattices-Hammersley-Clifford/ec75e3ca906681bd900218a348a4a35dfed3d6fd (visited on 07/05/2022).

Han, Yanjun; Jiao, Jiantao; Weissman, Tsachy (2016). "Minimax Rate-Optimal Estimation of KL Divergence between Discrete Distributions". In: *2016 International Symposium on Information Theory and Its Applications (ISITA)*. Oct. 2016, pp. 256–260.

– (2020). "Minimax Estimation of Divergences Between Discrete Distributions". In: *IEEE Journal on Selected Areas in Information Theory* 1.3 (Nov. 2020), pp. 814–823. ISSN: 2641-8770. DOI: 10.1109/JSAIT.2020.3041036.

Harries, Michael (1999). *SPLICE-2 Comparative Evaluation: Electricity Pricing*. Technical report. 1999.

Heggernes, Pinar (2006). "Minimal Triangulations of Graphs: A Survey". In: *Discrete Mathematics*. Minimal Separation and Minimal Triangulation 306.3 (Feb. 28, 2006), pp. 297–317. ISSN: 0012-365X. DOI: 10.1016/j.disc.2005.12.003.

Hellinger, E. (1909). "Neue Begründung der Theorie quadratischer Formen von unendlichvielen Veränderlichen." In: *Journal für die reine und angewandte Mathematik* 136 (1909), pp. 210–271. ISSN: 0075-4102. URL: http://www.digizeitschriften.de/dms/img/?PID=GDZPPN002166941 (visited on 02/02/2020).

Helly, Ed (1923). "Über Mengen konvexer Körper mit gemeinschaftlichen Punkte." In: *Jahresbericht der Deutschen Mathematiker-Vereinigung* 32 (1923), pp. 175–176. ISSN: 0012-0456; 1869-7135. URL: https://eudml.org/doc/145659 (visited on 03/05/2022).

Horn, W. A. (1972). "Three Results for Trees, Using Mathematical Induction". In: *Journal of Research of the National Bureau of Standards, Section B: Mathematical Sciences* 76B.1-2 (Jan. 1972), p. 39. ISSN: 0098-8979. DOI: 10.6028/jres.076B.002.

Ilić, Velimir M.; Djordjević, Ivan B.; Stanković, Miomir (2017). "On a General Definition of Conditional Rényi Entropies". In: *Proceedings* 2.4 (4 2017), p. 166. ISSN: 2504-3900. DOI: 10.3390/ecea-4-05030.

Jeffreys, Sir Harold (1998). *The Theory of Probability*. Third Edition, Third Edition. Oxford Classic Texts in the Physical Sciences. Oxford, New York: Oxford University Press, Aug. 6, 1998. 470 pp. ISBN: 978-0-19-850368-2.

Jiao, Jiantao; Han, Yanjun; Weissman, Tsachy (2018). "Minimax Estimation of the L1 Distance". In: *IEEE Transactions on Information Theory* 64.10 (Oct. 2018), pp. 6672–6706. ISSN: 1557-9654. DOI: 10.1109/TIT.2018.2846245.

Klinkenberg, Ralf (2001). "Using Labeled and Unlabeled Data to Learn Drifting Concepts". In: *In Workshop Notes of IJCAI-01 Workshop on Learning from Temporal and Spatial Data*. AAAI Press, 2001, pp. 16–24.

Koller, Daphne; Friedman, Nir (2009). *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009. ISBN: 978-0-262-01319-2.

Kullback, S.; Leibler, R. A. (1951). "On Information and Sufficiency". In: *The Annals of Mathematical Statistics* 22.1 (Mar. 1951), pp. 79–86. ISSN: 0003-4851, 2168-8990. DOI: 10.1214/aoms/1177729694.

Lauritzen, Steffen L. (1996). *Graphical Models*. Oxford Statistical Science Series 17. Oxford : New York: Clarendon Press ; Oxford University Press, 1996. 298 pp. ISBN: 978-0-19-852219-5.

Lauritzen, Steffen L.; Spiegelhalter, David J. (1988). "Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems".

In: *Journal of the Royal Statistical Society. Series B (Methodological)* 50.2 (1988), pp. 157–224.

MacKay, David J. C. (2003). *Information Theory, Inference and Learning Algorithms*. 1st edition. Cambridge, UK ; New York: Cambridge University Press, Sept. 25, 2003. 640 pp. ISBN: 978-0-521-64298-9.

Maniu, Silviu; Senellart, Pierre; Jog, Suraj (2019). "An Experimental Study of the Treewidth of Real-World Graph Data". In: *22nd International Conference on Database Theory (ICDT 2019)*. Ed. by Pablo Barcelo; Marco Calautti. Vol. 127. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhlâ€"Leibniz-Zentrum fuer Informatik, 2019, 12:1–12:18. ISBN: 978-3-95977-101-6. DOI: 10.4230/LIPIcs.ICDT.2019.12. URL: http://drops.dagstuhl.de/opus/volltexte/2019/10314 (visited on 02/02/2023).

Mezzini, Mauro; Moscarini, Marina (2010). "Simple Algorithms for Minimal Triangulation of a Graph and Backward Selection of a Decomposable Markov Network". In: *Theoretical Computer Science* 411.7 (Feb. 28, 2010), pp. 958–966. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2009.10.004.

Minku, Leandro L.; White, Allan P.; Yao, Xin (2010). "The Impact of Diversity on Online Ensemble Learning in the Presence of Concept Drift". In: *IEEE Transactions on Knowledge and Data Engineering* 22.5 (May 2010), pp. 730–742. ISSN: 1041-4347. DOI: 10.1109/TKDE.2009.156.

Moral, Serafín; Cano, Andrés; Gómez-Olmedo, Manuel (2021). "Computation of Kullback–Leibler Divergence in Bayesian Networks". In: *Entropy* 23.9 (9 Sept. 2021), p. 1122. DOI: 10.3390/e23091122.

Orlitsky, Alon; Santhanam, Narayana P.; Viswanathan, Krishnamurthy; Zhang, Junan (2004). "On Modeling Profiles Instead of Values". In: *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*. UAI '04. Arlington, Virginia, USA: AUAI Press, July 7, 2004, pp. 426–435. ISBN: 978-0-9749039-0-3.

Pavlichin, Dmitri S.; Jiao, Jiantao; Weissman, Tsachy (2019). "Approximate Profile Maximum Likelihood". In: *Journal of Machine Learning Research* 20.122 (2019), pp. 1–55. ISSN: 1533-7928. URL: http://jmlr.org/papers/v20/18-075.html (visited on 01/24/2021).

Pearl, Judea (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988. 552 pp. ISBN: 978-1-55860-479-7.

Pearl, Judea; Geiger, Dan; Verma, Thomas (1989). "Conditional Independence and Its Representations". In: *Kybernetika* 25.7 (1989), pp. 33–44. URL: http://www.kybernetika.cz/content/1989/7/33 (visited on 06/22/2022).

Pearson, Karl (1900). "X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 50.302 (1900), pp. 157–175. DOI: 10.1080/14786440009463897. eprint: https://doi.org/10.1080/14786440009463897. URL: https://doi.org/10.1080/14786440009463897.

Petitjean, François; Allison, Lloyd; Webb, Geoffrey I. (2014). "A Statistically Efficient and Scalable Method for Log-Linear Analysis of High-Dimensional Data". In: *2014 IEEE International Conference on Data Mining*. Shenzhen, China: IEEE, Dec. 2014, pp. 480–489. DOI: 10.1109/ICDM.2014.23.

Petitjean, François; Buntine, Wray; Webb, Geoffrey I.; Zaidi, Nayyar (2018). "Accurate Parameter Estimation for Bayesian Network Classifiers Using Hierarchical Dirichlet Processes". In: *Machine Learning* 107.8-10 (Sept. 2018), pp. 1303–1331. ISSN: 0885-6125, 1573-0565. DOI: 10.1007/s10994-018-5718-0.

Petitjean, François; Webb, Geoffrey I. (2015). "Scaling Log-Linear Analysis to Datasets with Thousands of Variables". In: *Proceedings of the 2015 SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, June 30, 2015, pp. 469–477. ISBN: 978-1-61197-401-0. DOI: 10.1137/1.9781611974010.53.

Petitjean, François; Webb, Geoffrey I.; Nicholson, Ann E. (2013). "Scaling Log-Linear Analysis to High-Dimensional Data". In: *2013 IEEE 13th International Conference on Data Mining*. Dallas, TX, USA: IEEE, Dec. 2013, pp. 597–606. ISBN: 978-0-7695-5108-1. DOI: 10.1109/ICDM.2013.17.

Poczos, Barnabas; Schneider, Jeff (2012). "Nonparametric Estimation of Conditional Information and Divergences". In: *Artificial Intelligence and Statistics*. PMLR, Mar. 21, 2012, pp. 914–923. URL: http://proceedings.mlr.press/v22/poczos12.html (visited on 01/24/2021).

Polikar, Robert; Upda, Lalita; Upda, Satish S.; Honavar, Vasant (2001). "Learn++: An Incremental Learning Algorithm for Supervised Neural Networks". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 31.4 (Nov. 2001), pp. 497–508. ISSN: 1094-6977. DOI: 10.1109/5326.983933.

Rahman, Mohammad S.; Haffari, Gholamreza (2020). "A Statistically Efficient and Scalable Method for Exploratory Analysis of High-Dimensional Data". In: *SN Computer Science* 1.2 (Feb. 7, 2020), p. 64. ISSN: 2661-8907. DOI: 10.1007/s42979-020-0064-2.

Rényi, Alfréd (1961). "On Measures of Entropy and Information". In: *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics* 4.1 (Jan. 1, 1961), pp. 547–562. URL: https://projecteuclid.org/ebooks/berkeley-symposium-on-mathematical-statistics-and-probability/Proceedings-of-the-Fourth-Berkeley-Symposium-on-Mathematical-Statistics-and/chapter/On-Measures-of-Entropy-and-Information/bsmsp/1200512181 (visited on 10/11/2022).

Schlimmer, Jeffrey C.; Granger, Richard H. (1986). "Incremental Learning from Noisy Data". In: *Machine Learning* 1.3 (1986), pp. 317–354. ISSN: 0885-6125, 1573-0565. DOI: 10.1007/BF00116895.

Scholz, Martin; Klinkenberg, Ralf (2005). "An Ensemble Classifier for Drifting Concepts". In: *In Proceedings of the Second International Workshop on Knowledge Discovery in Data Streams*. 2005, pp. 53–64.

Scutari, Marco (2010). "Learning Bayesian Networks with the Bnlearn R Package". In: *Journal of Statistical Software* 35 (July 16, 2010), pp. 1–22. ISSN: 1548-7660. DOI: 10.18637/jss.v035.i03.

Sebastião, Raquel; Gama, João (2007). "Change Detection in Learning Histograms from Data Streams". In: *Progress in Artificial Intelligence*. Ed. by José Neves; Manuel Filipe Santos; José Manuel Machado. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2007, pp. 112–123. ISBN: 978-3-540-77002-2. DOI: 10.1007/978-3-540-77002-2_10.

Sebastião, Raquel; Gama, João; Rodrigues, Pedro Pereira; Bernardes, João (2010). "Monitoring Incremental Histogram Distribution for Change Detection in Data Streams". In: *Knowledge Discovery from Sensor Data*. Ed. by Mohamed Medhat Gaber; Ranga Raju Vatsavai; Olufemi A. Omitaomu; João Gama; Nitesh V. Chawla; Auroop R. Ganguly. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2010, pp. 25–42. ISBN: 978-3-642-12519-5. DOI: 10.1007/978-3-642-12519-5_2.

Şeker, Oylum; Heggernes, Pinar; Ekim, Tınaz; Taşkın, Z. Caner (2017). "Linear-Time Generation of Random Chordal Graphs". In: *Algorithms and Complexity*. Ed. by Dimitris Fotakis; Aris Pagourtzis; Vangelis Th. Paschos. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 442–453. ISBN: 978-3-319-57586-5. DOI: 10.1007/978-3-319-57586-5_37.

– (2018). "Generation of Random Chordal Graphs Using Subtrees of a Tree". In: (Oct. 31, 2018). URL: http://arxiv.org/abs/1810.13326v1 (visited on 12/07/2021).

Sethi, Tegjyot Singh; Kantardzic, Mehmed (2017). "On the Reliable Detection of Concept Drift from Streaming Unlabeled Data". In: *Expert Systems with Applications: An International Journal* 82.C (Oct. 1, 2017), pp. 77–99. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2017.04.008.

Shaker, Ammar; Hüllermeier, Eyke (2015). "Recovery Analysis for Adaptive Learning from Non-Stationary Data Streams: Experimental Design and Case Study". In: *Neurocomputing* 150 (Feb. 2015), pp. 250–264. ISSN: 09252312. DOI: 10.1016/j.neucom.2014.09.076.

Shannon, C. E. (1948). "A Mathematical Theory of Communication". In: *The Bell System Technical Journal* 27.3 (July 1948), pp. 379–423. ISSN: 0005-8580. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

Sibson, Robin (1969). "Information Radius". In: *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete* 14.2 (June 1, 1969), pp. 149–160. ISSN: 1432-2064. DOI: 10.1007/BF00537520.

Street, W. Nick; Kim, YongSeog (2001). "A Streaming Ensemble Algorithm (SEA) for Large-scale Classification". In: *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '01. New York, NY, USA: ACM, 2001, pp. 377–382. ISBN: 978-1-58113-391-2. DOI: 10.1145/502512.502568.

Teixeira, Andreia; Matos, Armando; Antunes, Luís (2012). "Conditional Rényi Entropies". In: *IEEE Transactions on Information Theory* 58.7 (July 2012), pp. 4273–4277. ISSN: 1557-9654. DOI: 10.1109/TIT.2012.2192713.

Tsymbal, Alexey (2004). *The Problem of Concept Drift: Definitions and Related Work*. Technical Report TCD-CS-2004-15. Computer Science Department: Trinity College Dublin, May 29, 2004.

Valiant, Leslie Gabriel (1979). "The complexity of enumeration and reliability problems". In: *SIAM Journal on Computing* 8.3 (1979), pp. 410–421.

Wainwright, Martin J.; Jordan, Michael I. (2008). "Graphical Models, Exponential Families, and Variational Inference". In: *Foundations and Trends in Machine Learning* 1.1–2 (2008), pp. 1–305.

Wang, Jindong; Lan, Cuiling; Liu, Chang; Ouyang, Yidong; Qin, Tao; Lu, Wang; Chen, Yiqiang; Zeng, Wenjun; Yu, Philip (2022). "Generalizing to Unseen Domains: A Survey on Domain Generalization". In: *IEEE Transactions on Knowledge and Data Engineering* (2022), pp. 1–1. ISSN: 1558-2191. DOI: 10.1109/TKDE.2022.3178128.

Webb, Geoffrey I.; Hyde, Roy; Cao, Hong; Nguyen, Hai Long; Petitjean, François (2016). "Characterizing Concept Drift". In: *Data Mining and Knowledge Discovery* 30.4 (July 2016), pp. 964–994. ISSN: 1384-5810, 1573-756X. DOI: 10.1007/s10618-015-0448-4. arXiv: 1511.03816.

Webb, Geoffrey I.; Lee, Loong Kuan; Goethals, Bart; Petitjean, François (2018). "Analyzing Concept Drift and Shift from Sample Data". In: *Data Mining and Knowledge Discovery* 32.5 (Sept. 1, 2018), pp. 1179–1199. ISSN: 1573-756X. DOI: 10.1007/s10618-018-0554-1.

Webb, Geoffrey I.; Petitjean, François (2016). "A Multiple Test Correction for Streams and Cascades of Statistical Hypothesis Tests". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*. San Francisco, California, USA: ACM Press, 2016, pp. 1255–1264. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939775.

Widmer, Gerhard; Kubat, Miroslav (1996). "Learning in the Presence of Concept Drift and Hidden Contexts". In: *Machine Learning* 23.1 (Apr. 1, 1996), pp. 69–101. ISSN: 0885-6125, 1573-0565. DOI: 10.1007/BF00116900.

Yannakakis, Mihalis (1981). "Computing the Minimum Fill-In Is NP-Complete". In: *SIAM Journal on Algebraic Discrete Methods* 2.1 (Mar. 1981), pp. 77–79. ISSN: 0196-5212. DOI: 10.1137/0602010.

Zhang, Wei; Yu, Yao-Chi; Li, Jr-Shin (2019). "Dynamics Reconstruction and Classification via Koopman Features". In: *Data Mining and Knowledge Discovery* 33.6 (Nov. 1, 2019), pp. 1710–1735. ISSN: 1573-756X. DOI: 10.1007/s10618-019-00639-x.

Zhang, Zhiyi; Grabchak, Michael (2014). "Nonparametric Estimation of Küllback-Leibler Divergence". In: *Neural Computation* 26.11 (Nov. 1, 2014), pp. 2570–2593. ISSN: 0899-7667. DOI: 10.1162/NECO_a_00646.

Zhou, Kaiyang; Liu, Ziwei; Qiao, Yu; Xiang, Tao; Loy, Chen Change (2022). "Domain Generalization: A Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022), pp. 1–20. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2022.3195549.

Zimek, Arthur; Schubert, Erich; Kriegel, Hans-Peter (2012). "A Survey on Unsupervised Outlier Detection in High-Dimensional Numerical Data". In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 5.5 (2012), pp. 363–387. ISSN: 1932-1872. DOI: 10.1002/sam.11161.

Žliobaite, Indre (2010). "Change with Delayed Labeling: When Is It Detectable?" In: *2010 IEEE International Conference on Data Mining Workshops*. Dec. 2010, pp. 843–850. DOI: 10.1109/ICDMW.2010.49.