FRIEDRICH-SCHILLER-
UNIVERSITÄT
JENA

Exploiting Markov Equivalence for Fast Inference

MASTER THESIS

Submitted for the degree of
MASTER OF SCIENCE (M.SC.)

FRIEDRICH SCHILLER UNIVERSITY JENA

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

MATHEMATICS

Author
Jenette SELLIN

Born December 22, 1995
in CALIFORNIA, USA

Advisor
Andreas GORAL

Supervisor
Prof. Dr. Joachim GIESEN

Jena, June 4, 2020

Abstract

As observational data collection becomes more accessible, the importance of modeling causality and answering inference queries efficiently increases. Causal relationships are often encoded in directed acyclic graphs (DAGs) and associated probability distributions, which together comprise a Bayesian network. Bayesian networks have the property of non-identifiability, meaning that several structurally distinguishable networks can encode the same probability distribution. Resultingly, the same inference queries can be asked over different Bayesian networks while producing statistically indistinguishable results. This thesis explores whether it can be computationally beneficial to exploit this non-identifiability property for faster inference; that is, whether one can achieve speedup in answering a sequence of inference queries by changing the representation of a Bayesian network on the fly to serve the queries. This exploration yields a short survey of related topics, a theorem for identifying the set of vertices a query depends on, an algorithm for making beneficial network transformations, and a discussion of their efficacy.

Keywords: Bayesian Networks, causal models, inference queries, optimization, non-identifiability, Markov Equivalence.

Zusammenfassung

Schlagwörter:

Acknowledgements

Notation and Abbreviations

DAG	Directed Acyclic Graph
iff	If and only if
$G = (V, E)$	A graph G with vertices V and edges E
$G, G', G_1, G_2, ..., G_n$	directed graphs
$H, H', H_1, H_2, ..., H_n$	undirected and mixed graphs
$B, B', B_1, B_2, ..., B_n$	Bayesian networks
Capital letters e.g. X, Y, Z	random variables
Lowercase letters e.g. x, y, z	values attained by random variables
$p(X = x)$	The probability that the random variable X takes on the value x.
$p(X = x, Y = y)$	The probability of $X = x$ and $Y = y$ simultaneously.
$p(X = x \mid Y = y)$	The probability that $X = x$ given the condition $Y = y$.
$p(X)$	The joint probability function of possible values attained by X.
$\alpha(X)$	the set of ancestors of a vertex X in a graph G
$\prod_X^G$	the set of parents of a vertex X in a graph G
$\Delta(G', q)$	the number of vertices involved in a query q on a graph G.
$\Delta*$	$\arg\min_{G' \in [G]}(\Delta(G', q))$.
$[G]$	The Markov equivalence class of G
MEC	Markov Equivalence Class
$G \approx_M G'$	G is Markov equivalent to G'
G_*	essential graph of G

Contents

Chapter 1

Introduction

1.1 Motivation

Many fields rely on causal models as a framework for drawing inference on observational data. Causal inference typically involves answering inference queries over a Bayesian network, that is, a directed acyclic graph (DAG) paired with an associated joint probability distribution. Computing inference queries is the process of reducing one probability distribution (typically via *conditioning* and *marginalizing*) into another simplified probability distribution from which we can draw samples, yielding a most probable configuration of the random variables involved.

For instance, in the medical field, we may have data about a large number of patients (such as age, sex, and blood type) and we may wish to predict whether the patient has a specific disease or not. In this scenario, one would represent the causal data as a Bayesian network and query the model for the likelihood that a patient has the disease. This amounts to evaluating the Bayesian network having observed the patient's data, and sample the resulting reduced probability distribution. This serves as the inference process, and allows for informed decision making.

The models discussed in this thesis have significant applications and adaptions in medicine [1], [2], [3], [4], [5], behavioral and social sciences [6], [7], and statistics [8], [9]. Furthermore, significant theoretical research has been done on causal models themselves [10] [11], [12]. An extensive survey of their genesis, development, and applications can be found in [13].

Naturally, especially as the use of statistical inference becomes increasingly widespread, it is in our best interest to explore methods for modeling and query-

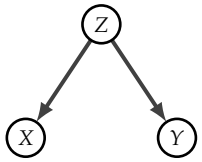
ing which allow us to do inference as quickly and efficiently as possible. This thesis therefore take advantage of the Bayesian networks' property of non-identifiability, meaning that there can be several networks which encode the same probability distribution. Such networks are called *Markov equivalent*. Markov equivalence informs us that a single inference query evaluated over structurally distinguishable networks can, in certain cases, yield in statistically indistinguishable results.

We therefore explore the following question: given that the same probability distribution can be encoded in different Bayesian networks, **can it be computationally beneficial to change the representation of the probability distribution in order to answer a sequence of queries more efficiently?**

Furthermore, in certain settings such as probabilistic programming, one does not evaluate an inference query over a probability distribution only once, but may instead wish to evaluate hundreds if not thousands or queries. Repeated evaluation of queries is a significant motivation for this work, as it justifies the overhead costs of transforming the networks.

1.2 Background

The family of models explored in this thesis are Bayesian networks over discrete probability distributions. The purpose of Bayesian networks is to encode information about vectors of random variables, namely the interdependencies satisfied by the vector's joint probability function. For example, consider a vector of three random binary variables (X, Y, Z). Suppose that the values of X and Y depend on Z, but X and Y are independent from one another given Z. These dependencies can be encoded in the graph in Figure 1.2.1, which is specified by its joint probabilisty function.

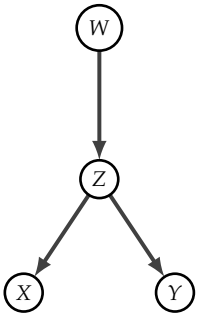


$$p(X, Y|Z) = p(X|Z) \cdot p(Y|Z)$$

Figure 1.2.1

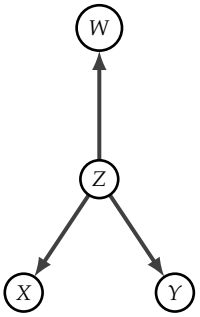
In the case of the directed models we explore in this thesis, there is not necessarily a unique graph which satisfies the dependency constraints of the joint probability function. A random vector and its constraints may be represented by a variety of different graph structures which ultimately describe the same underlying probability distribution. To understand how the complexity of answering a query depends on graph structure, consider the two models in Figure 1.2.2.

If we wish to answer a query about the random variable Y in G_1, for instance



G_1

$$p(X,Y,Z,W) = p(X|Z) \cdot p(Y|Z) \cdot p(Z|W) \cdot p(W)$$



G_2

$$p(X,Y,Z,W) = p(X|Z) \cdot p(Y|Z) \cdot p(W|Z) \cdot p(Z)$$

Figure 1.2.2

the probability that Y attains the value y, notated $p(Y = y)$, we must also compute the probabilities of each vertex it depends on either directly or indirectly: in this case, all of its **ancestors** (namely Z and W) in addition to Y itself. Alternatively, if we wish to answer a query $p(Y = y)$ graph G_2, we need only to compute Z and Y. Then the number of variables which must be computed is reduced when we consider G_2 instead of G_1. The set of vertices a query depends on can somewhat vary depending on the query, but the motivation for transforming a network remains the same.

In a context where answering queries about a single vertex may require us to compute arbitrarily many other vertices in the graph, we seek an equivalent graphical representation of our probability distribution which minimizes the number of vertices that must be computed when answering a query. Further, we wish to explore the optimal sequence of tranformations allowing us to answer a sequence of queries most efficiently.

1.3 Goal

The central concept of this thesis is the exploitation of non-identifiability of Bayesian networks (Markov equivalence) for faster inference. Given that multiple Bayesian networks may encode the same probability distribution, the goals of our research are as follows:

Given an inference query q and a Bayesian network B (which represents a single factorization of the probability distribution we wish to query), find a Markov equivalent Bayesian network B' which minimizes the number of variables that must be evaluated to answer the query. The purpose of the minimization is to reduce the cost of evaluating q in order to achieve faster inference, as well as evaluate the gained speed-up versus the cost of transforming the representation.

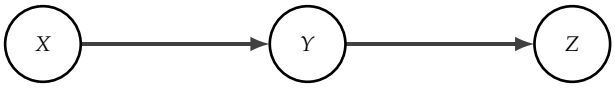
To achieve this, we first define the set of nodes which must be evaluated to answer q over a given network B, notated $\Delta(B,q)$. Then, we search for a Markov equivalent Bayesian network B' such that the size of $\Delta(B',q)$ is minimized. Finally, we compare the costs of computing the query over B to the costs of computing the query over the minimized graph B' plus the costs of identifying and transforming to B'. In short, the minimization problem is to determine the network Δ^* defined as

$$\Delta^* = argmin_{B' \in [B]} \Delta(B',q).$$

Subsequently, we consider how to identify optimal transformations for a known sequence of queries $\{q_0, q_1, ...q_n\}$ such that the overhead of transformation is most beneficial compared to the gained speedups by transforming.

1.4 Motivating Example

The following example demonstrates how answering a query on one graph can be sped up by asking the same query over a Markov equivalent graph with a different structure. Consider the DAG in Figure 1.4.1.



G_1

Figure 1.4.1

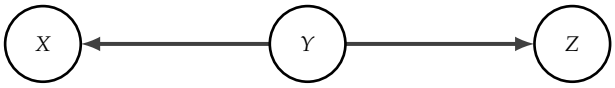
with conditional probabilities given by

$$p(X=1)=0.2$$
$$p(Y=1|X=0)=0.3 \qquad p(Y=1|X=1)=0.4$$
$$p(Z=1|Y=0)=0.1 \qquad p(Z=1|Y=1)=0.9$$

and the joint probability

$$p(X,Y,Z)=p(X)\cdot p(Y|X)\cdot p(Z|Y)$$

Here, if we wish to answer the query $p(Z|Y)$, our computation relies on all three random variables, since Z relies on Y and Y relies on X. However, we can find a Markov equivalent graph in which we can answer the same query while relying on fewer random variables. Consider the DAG in Figure 1.4.2.



G_2

Figure 1.4.2

We compute the conditional probabilities of G_2 to show that the model can equivalently describe our joint probability distribution from G_1, and therefore be used to answer the query $p(Z|Y)$. We then observe how the structure of the new model affects our computation of the query. By applying Bayes' Theorem, which states that $p(A|B)=\frac{p(B|A)p(A)}{p(B)}$, we can compute

$$p(X|Y)=\frac{p(Y|X)\cdot p(X)}{p(Y)}=\frac{p(Y|X)\cdot p(X)}{\sum_X p(Y|X)\cdot p(X)}$$

Using our pre-defined conditional probabilities,

$$\begin{aligned} p(Y=0) &= \sum_X p(Y=0|X) \cdot p(X) \\ &= 0.7 \cdot 0.8 + 0.6 \cdot 0.2 \\ &= 0.56 + 0.12 \\ &= 0.68 \end{aligned}$$

and

$$\begin{aligned} p(Y=1) &= \sum_X p(Y=1|X) \cdot p(X) \\ &= 0.3 \cdot 0.8 + 0.4 \cdot 0.2 \\ &= 0.24 + 0.08 \\ &= 0.32. \end{aligned}$$

which allow us to compute

$$p(X=1|Y=0) = \frac{p(Y=0|X=1) \cdot p(X=1)}{p(Y=0)} = \frac{0.6 \cdot 0.2}{0.68} = \frac{3}{17}$$

and

$$p(X=1|Y=1) = \frac{p(Y=1|X=1) \cdot p(X=1)}{p(Y=1)} = \frac{0.4 \cdot 0.2}{0.32} = \frac{1}{4}.$$

Then, since X and Y do not depend on Z, the conditional probability of Z remains unchanged. Therefore, model G_2 is an equivalent model to G_1. If we answer the query $p(Z|Y)$ on model G_2, we do not need to consider X, since Y (and consequently Z) no longer depends on X.

Through this example, we see that it may be possible to find an equivalent graph to our original model, and to use the new graph structure to answer certain queries more efficiently. It is important to note that the efficiency of the new model for answering a query relies both on the conditional probabilites we are interested in, as well as the content of our query (as opposed to an arbitrary query).

1.5 Related work

Significant work has been done on individual aspects of the subject, though this thesis pioneers in using them together for fast inference.[10] give a framework for understanding how the same probability distribution can be encoded into two distinct graphs, as well as present a simplified criterion for when two graphs describe indistuingishable distributions. [14] present a survey-style overview of Markov Equivalent graph structures, reduction of graph structures to simplified representations of equivalent distributions, as well as providing a strong background of relevant graph theory and probability preliminaries. [11] explores parameters of equivalent networks, presents a procedure for altering graphs without changing the described probability distribution, and quantifies the complexity of several such manipulations. [12] make similar headway by exploring reductions of graphs to representative form, as well as describing procedures by which one can alter the structure.

Adjacently, [15] confronts the same goal of faster inference of queries using a different method in a modified setting. His task focuses on eliminating unnecessary nodes from the computations directly, rather than by reversing edges.

1.6 Outline

In Chapter 2, we introduce preliminary concepts from graph theory and Bayesian statistics to robustly define our models and to aid our understanding of graph manipulations and the process of querying over a graph. Chapter 3 details the problem and proposed querying method, then aims to quantify the computational costs of answering an arbitrary query over a graph by this method. Chapter 4 builds context for understanding Markov equivalence between graphs. In Chapter 5, we examine the circumstances under which a Markov equivalence can be utilized to answer a query more efficiently, and then quantify the computational costs of finding a suitable Markov equivalent graph for faster inference. Finally, in Chapter 6, we compare the quantities explored in Chapters 3 and 5 to determine whether our proposed algorithm indeed increases inference speed, and if so, where it is effective.

Chapter 2

Preliminaries

In this chapter we introduce preliminary concepts from graph theory and Bayesian statistics. These concepts will allow us to robustly define our models and problem statements, give us critial background for evaluating queries and exploring Markov Equivalence, and serve as a basis for the content of the rest of the thesis.

2.1 Foundations of Graph Theory

Definition 2.1.1 (Graph (directed, undirected, mixed)). *A **graph** is defined as a pair $G = (V, E)$ in which V is a finite set of vertices, and $E \subset V \times V$ is a finite set of edges. Vertices (also sometimes called **nodes**) will generally be denoted by capital letters, i.e. X, Y, Z. G is called **undirected** if every edge in G is undirected, meaning that for each edge $(X, Y) \in E$ (denoting an edge from vertex X to vertex Y), the edge (Y, X) is also in E, for $X \neq Y$. Otherwise, G is called **directed**. We denote such undirected edges by $\{X, Y\}$. A **mixed graph** (also called a hybrid graph) is a graph G which contains both directed and undirected edges.*

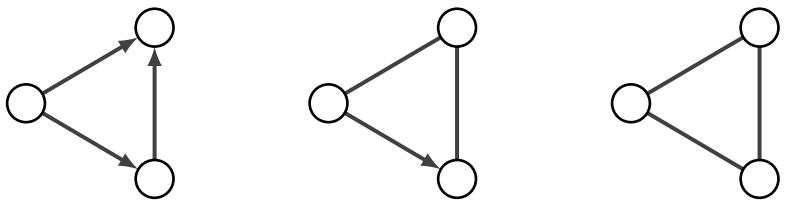


Figure 2.1.1

Definition 2.1.2 (Neighbor, adjacent). *Two vertices $X, Y \in V$ in a graph $G = (V, E)$ are called **adjacent** if there is an edge between them, either (X, Y), (Y, X), or X, Y. X and Y are also called **neighbors** in this scenario.*

Definition 2.1.3 (Route). *A **route** in a graph $G = (V, E)$ is a sequence $X_1, X_2, ..., X_k$ of vertices in V such that there is an edge connecting X_i to X_{i+1} (independent of the direction of the edge), for $i = 1, ..., k-1, k \geq 1$. The integer k is the length of the route.*

Definition 2.1.4 (Path, directed cycle). *A (directed) **path** in a directed graph $G = (V, E)$ is a route where vertices X_i and X_{i+1} are connected by a directed edge (X_i, X_{i+1}). A directed path which begins and ends at the same vertex is called a **directed cycle**.*

Remark. The distinction between a path and a route will be significant in later chapters. One should retain that a path is a directed sequence while a route is undirected.

Definition 2.1.5 (Parent, ancestor). *Given two vertices $X, Y \in V$ in a directed graph $G = (V, E)$, Y is called a **parent** of X if there is an edge $(Y, X) \in E$. The set of parents of a vertex X in a graph is denoted $\prod_X^G$. Likewise, X is called a **child** of Y. The set of **ancestors** of X, denoted $\alpha(X)$, is the set of all vertices Y such that there exists a directed path from Y to X, but no path from X to Y. In the context of this thesis, this is the set of vertices upon which a vertex X depends, either directly or indirectly. Likewise, if Y is an ancestor of X, then X is a **descendant** of Y.*

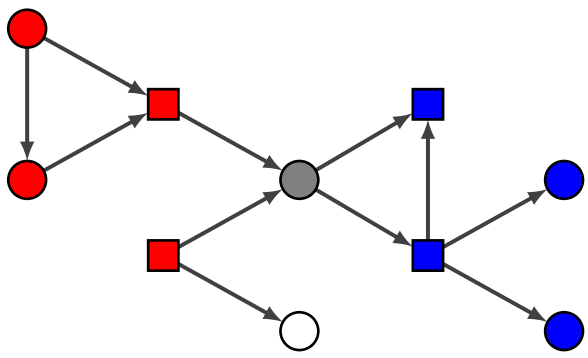


Figure 2.1.2: Red vertrices are ancestors of the gray vertex (squares denoting parents), and blue vertices are descendants of it (squares denoting children). The white vertex is neither an ancestor nor a descendent of the gray node.

Remark. $\prod_X^G \subseteq \alpha(X)$. Likewise, the set of children of X is a subset of the set of descendents of X.

Definition 2.1.6 (Chain graph, directed acyclic graph (DAG)). *A mixed graph $G = (V, E)$ is called a **chain graph** if it contains no directed cycles. A **directed acyclic graph** is a chain graph which is directed. This is abbreviated as DAG.*

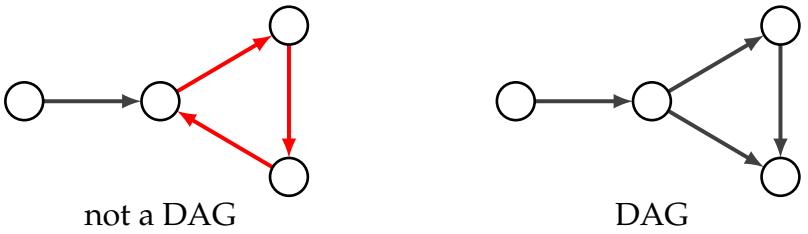


Figure 2.1.3: A directed cycle shown in red.

Remark. Every undirected graph is a chain graph.

2.2 Foundations of Bayesian Statistics

Definition 2.2.1 (Bayesian Network [16]). *A **Bayesian network** is a pair $B = (G, P)$ where $G = (V, E)$ is a DAG and P is a joint probability distribution defined on a set of random variables $\mathcal{X}$. That is, it is a DAG and an associated joint probability distribution.*

Vertices in G have a one-to-one correspondence to the random variables in $\mathcal{X}$ associated with P, meaning the set of vertices V represents both the vertices of G and the random variables of the joint probability distribution P.

P factorizes on the sets $\{X \cup \prod_X^G | X \in G\}$. That is, for each $X \in G$ there is a factor in P that depends on X and $\prod_X^G$, namely $P_X(X|\prod_X^G)$.

Example 2.2.1 (Simple Bayesian Network). The DAG in Figure 2.2.1 paired with the joint probability distribution $p(X_1 = x_1, X_2 = x_2) = p(X_1 = x_1) \cdot p(X_2 = x_2 | X_1 = x_1)$ is a Bayesian network.

Figure 2.2.1

Remark. In the context of this thesis, we only consider graphs G which are DAGs unless explicitly stated otherwise. Further, for our purposes, all such DAGs are directed graphical models with associated probability distributions. Therefore we will refer to DAGs $G = (V, E)$ and Bayesian networks interchangably, with necessary details clarified in context.

Definition 2.2.2 (Sample space, event [17]). *Given a random experiment (an experiment whose outcome depends on chance), the* ***sample space*** *of the experiment is the set of all possible outcomes. Each subset of a sample space is called an* ***event****, that is, an outcome or a collection of outcomes of the random experiment. Therefore, event* $\subseteq$ *sample space.*

Example 2.2.2. In the context of Bayesian networks, we are interested in possible values of random variables which are represented by vertices. As a simple example, suppose we have the random binary variables X and Y. Then the sample space of X is $\{0,1\}$, and likewise for Y. Generally, we will be interested in answering questions such as the probability that the random variable X takes on a value $x \in \{0,1\}$; these queries are written $p(X = 0)$, $p(X = 1)$, or more generally, $p(X = x)$. The query $p(X = x)$ phrased mathematically as: compute

$$argmax_{x \in \{0,1\}} p(X = x).$$

Some possible queries one might encounter include the following:

- $p(X = x, Y = y)$, the probability that $X = x$ and $Y = y$ simultaneously, computed $p(X = x) \cdot p(Y = y)$,
- $p(X = x | Y = y)$, the probability that $X = x$ given that we have observed the condition $Y = y$.

Let the joint probabilities of these random vectors be given as follows:

$$p(X = 1) = \frac{3}{10} \qquad p(X = 0) = \frac{7}{10}$$
$$p(Y = 0) = \frac{6}{10} \qquad p(Y = 1) = \frac{4}{10}$$

where for X and Y binary variables, $p(X = 1) = 1 - p(X = 0)$ since $\sum_{x \in \{0,1\}} p(X = x) = 1$. This joint probability function $p : \{0,1\} \times \{0,1\} \to [0,1]$ can also be encoded as a table:

X	1	1	0	0
Y	1	0	1	0
$p(X,Y)$	$\frac{3}{25}$	$\frac{21}{100}$	$\frac{7}{25}$	$\frac{21}{50}$

So, the joint probability function $p(X,Y)$ is not a value, but a function of the likelihood of attainable values from the sample space of Z and Y. Then, we can answer the query $p(X=1,Y=0)$ by reading off the table: $\frac{3}{25}$.

Next, suppose we have observed the condition $Y=0$. Then, we can *condition* the distribution, meaning we update the probability function such that it reflects our observation. We do so by looking at the scenarios in which $Y=0$:

$$p(X=0|Y=0)=\frac{21}{50} \qquad \text{and} p(X=1|Y=0)=\frac{21}{100}.$$

Then, using the fact that the probabilities should sum up to 1, we normalize:

$$\begin{aligned} p(X=0|Y=0) &= \frac{p(X=0|Y=0)}{p(X=0,Y=0)+p(X=1,Y=0)} \\ &= \frac{p(X=1,Y=0)}{p(Y=0)} \end{aligned}$$

and

$$\begin{aligned} p(X=1|Y=0) &= \frac{p(X=1,Y=0)}{p(X=0,Y=0)+p(X=1,Y=0)} \\ &= \frac{p(X=1,Y=0)}{p(Y=0)} \end{aligned}$$

Thus, with $p(Y=0)=\frac{63}{100}$, these give us

$X\|Y=1$	1	0
$p(X\|Y=1)$	$\frac{2}{3}$	$\frac{1}{3}$

Definition 2.2.3 (Probabilistic query). *A probabilistic query q on a graph $G=(V,E)$ is an inference question of the form $p(X_1,...,X_n|Y_1,...,Y_m)$ where $X_i, i\in[0,n]\in V$ and $Y_j, j\in[0,m]\in V$. X_i are the **targets** of q while Y_j are the **conditions** or **observations**.*

Theorem 2.2.1 (Bayes' Theorem [17]). *Given two random variables X and Y with $p(Y) \neq 0$*

$$p(X|Y) = \frac{p(Y|X) \cdot p(X)}{p(Y)}.$$

Remark. Bayes' Theorem will serve as a significant basis for many of the calculations necessary for transforming graphs in later chapters.

Definition 2.2.4 ((Statistical) independence [17]). *Let X and Y be random variables. Then X and Y are called **independent** whenever*

$$P(X, Y) = p(X)p(Y)$$

or equivalently if

$$p(X) = p(X|Y)$$

and vice versa.

Remark. 2.2.4 can be intuitively understood as follows: learning about Y has no effect on our knowledge concerning X and vice versa.

Definition 2.2.5 (Conditional independence [17]). *Let X, Y, and Z be random variables. Then X is said to be **conditionally independent** of Y given Z iff $p(X|Y, Z) = p(X|Z)$. Otherwise, they are said to be **conditionally dependent**.*

Remark. Definition 2.2.5 can be intuitively understood as follows: If we know about Z, then learning about Y has no effect on our expectation concerning X. Likewise, learning about X has no effect on our expectation concerning Y if we know about Z.

Remark. Definition 2.2.4 and Definition 2.2.5 also hold for disjoint sets A, B, and $C \subset V$ with p a joint probability distribution defined on V.

Example 2.2.3. Suppose Evin and Lisa take turns flipping a single coin which either results in heads (H) or tails (T). Naturally, one would assume that the probability of Lisa flipping heads is independent of the probability that Evin flips heads, that is,

$$p(Lisa = H|Evin = H) = p(Lisa = H),$$

since flipping a fair coin once does not dictate the outcome of a second flip. This means the two experiments are independent.

Now suppose we learn that the the coin is biased, calling this event Z. If the first coin flip results in heads, we would guess that the coin is more likely biased toward heads, and expect that the second coin flip to be heads. That is, the event of Evin flipping heads gives us information about the likelihood that Lisa will flip heads, given that we have observed Z. Then, whereas $p(Evin = H)$ and $p(Lisa = H)$ were previously independent from one another, observing event Z makes them depend on one another. Therefore the two events are conditionally dependent given Z:

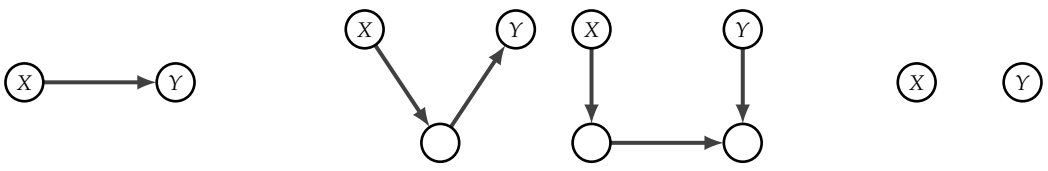
$$p(Lisa = H|Z, Evin = H) \neq p(Lisa = H|Evin = H).$$

Dependence, independence, and conditional independence have an intuitive manifestation within graph structure, which follows from Definition 2.2.4.

Given a graph $G = (V, E)$ such as any of the graphs in Figure 2.2.2, if two nodes $X, Y \in V$ have an edge between them, either (X, Y) or (Y, X), then they are dependent variables by construction of our graphs.

If there exists a route between two nodes X and Y, but X and Y are not adjacent, then they are conditionally independent, where the conditions are the nodes on that route. This is because information about X can give us information about an intermediary node on the route and allow us to gain information about Y (or vice versa).

Finally, if two nodes are not connected by a route (disconnected) then they are independent, since information about X cannot give us information about Y under any circumstances.



dependent X and Y conditionally independent independent

Figure 2.2.2

Chapter 3

Querying

3.1 Outline of the Task

The first task outlined by our goal is to determine the set of random variables that must be involved in the computation to answer a given query over G. This will serve as our cost function for the minimization which occurs in later chapters; the minimization will be to reduce the cost, meaning to find a Markov equivalent graph G' to G such that this number of vertices involved is minimized.

Naturally, before we consider the minimization itself, we begin by identifying such a set of vertices. This chapter will focus on identifying this set of vertices which must be computed to answer a query.

3.2 Structure of a Query

An inference query is a question which askes the probability of an event given some observations. Queries can take advantage of multiple observations and have multiple targets. For example, one can ask for the probability that the random variable X has takes on the value x given that we have observed $Y = y$. Let x be a value attained by the random variable X, and let y be a value attained by the variable Y. This is written $p(X = x|Y = y)$. The general form of a query is as follows:

Definition 3.2.1 (Query). *Given a graph $G = (V, E)$ with $X_1, ...X_n, Y_1, ...Y_n \in V$, the general form of a query q is $p(X_1, ..., X_n|Y_1, ...Y_n)$, meaning the probability of variables $X_1, ..., X_n$ given that we have conditioned on observed values attained by variables $Y_1, ..., Y_n$.*

Remark. Note that while a query on $G = (V, E)$ must have at least one target (otherwise there is nothing to compute), it does not necessarily need to include a condition. For example, $p(X = x)$ is a valid query for $X \in V$.

Scenario	Query	Interpretation	Example
Query a single target, single observation	$p(X = x \mid Z = z)$	Probability that $X = x$ given that we have observed $Z = z$.	Probability that a patient has lung cancer given that they are a tobacco smoker.
Query with multiple targets, single observation	$p(X = x, Y = y \mid Z = z)$	Probability that $X = x$ and $Y = y$ given that we have observed $Z = z$.	Probability that a patient has lung cancer and high blood pressure given that they are a tobacco smoker.
Query with a single target, multiple observations	$p(X = x \mid W = w, Z = z)$	Probability that $X = x$ given that we have observed $W = w$ and $Z = z$.	Probability that a patient has lung cancer given that they are over 50 years of age and a tobacco smoker
Query with multiple targets, multiple observations	$p(X = x, Y = y \mid W = w, Z = z)$	Probability that $X = x$ and $Y = y$ given that we have observed $W = w$ and $Z = z$.	Probability that a patient has lung cancer and high blood pressure given that they are over 50 years of age and a tobacco smoker.

Let G be a DAG and q a query. Define $\Delta(G, q)$ to be the number of nodes involved in the computation of q on G. The goal of this chapter is to find this value by determining the members of the set. Ultimately, we intend to find $\arg\min_{G' \in [G]}(\Delta(G, q))$. We call this minimized graph Δ^*.

3.3 Cost of a Query

In the following section, we determine which sets of vertices a query q on a DAG G depends on, notated $\Delta(G, q)$. First we consider queries with single targets and single conditions, then move on to scenarios with multiple conditions, before finally generalizing to scenarios with multiple targets and conditions. We will see that the structue of the DAG (namely, position of targets in G in relation to the position of conditions) plays a significant role in whether certain vertices must be considered to compute a query.

Determining $\Delta(G, q)$ for an arbitrary q will later allow us to quantify the speedups gained by transforming the graph: given a DAG G and a Markov-equivalent transformed DAG G', will compare the $\Delta(G, q)$ to $\Delta(G', q)$. If $\Delta(G', q)$ is a smaller set, then one can conclude that the transformation has reduced the

cost of the query. Then the question becomes whether the overhead of transformation is worth the earned speedup. Once again, we will take advantage of the definition of conditional independence:

$$P(A|B) = \frac{P(A,B)}{P(B)}.$$

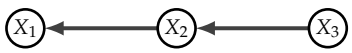
Lemma 3.3.1. *Computing a conditionless query $q = p(X_0, X_1, ..., X_n)$ over a DAG G depends on vertices $X_0, X_1, ..., X_n$ and all of their ancestors: $\alpha(X_0), \alpha(X_1), ..., \alpha(X_n)$.*

Proof. Let $q = p(X_0, X_1, ..., X_n)$ on a DAG G. Using the definition of a Bayesian network, we have that the joint probability distribution

$$p(X_0, X_1, ...X_n) = \prod_{i=0}^{n} p(X_i | \prod_{X_i}^{G}).$$

Therefore q indeed depends on all ancestors of the targets. □

Example 3.3.1. Let $q = p(X_1 = x_1)$ be a query on $G = (V, E)$ in Figure 3.3.1. The computation of q is as follows.



G

Figure 3.3.1

$$p(X_1 = x_1) \;=\; \sum_{x_2} \sum_{x_3} p(X_1 = x_1 | X_2 = x_2) \cdot p(X_2 = x_2 | X_3 = x_3) \cdot p(X_3 = x_3)$$

and therefore requires considering the target vertex X_1 and its ancestors, namely X_2 and X_3.

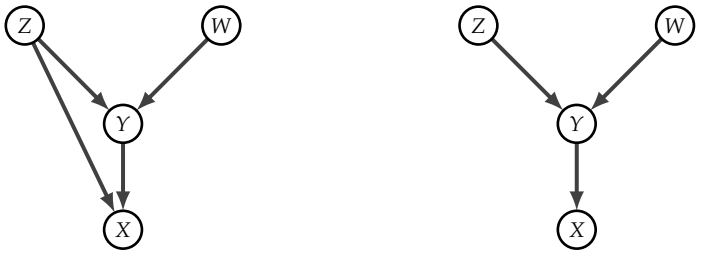
Theorem 3.3.2. *Let $q = p(X|Y)$ be a query on a DAG G. The the computation of q requires the following sets of vertices depending on the position of Y in G:*

- ***Special case** If X is conditionally independent of all ancestors of Y given Y, then q depends on X, $\alpha(X) \backslash \alpha(Y)$, and all nodes on a route from X to Y excluding Y itself.*

- ***General case*** *Otherwise, in the case that X is* ***not*** *independent of $\alpha(Y)$ given Y, q depends on both X and Y in addition to the sets $\alpha(X)$ and $\alpha(Y)$, as well as all nodes on a route from X to Y.*

Remark. The conditions for the cases in 3.3.2 can be rephrased as follows:

- The condition Y is in the **special case** if, when Y is removed from G, the resulting graph G' consists of at least two disconnected subgraphs G'_1 and G'_2 such that $X \in G'_1$ and all vertices in $\alpha(Y) \in G'_2$. That is, removing Y disconnects $\alpha(Y)$ from X.
- The condition Y is in the **general case** otherwise. That is, whenever the removal of Y does not result in a disconnected graph such that X and all vertices in $\alpha(Y)$ are in separate subgraphs.



General case Special case

Figure 3.3.2: Example of one DAG in the general case and one in the special case.

Proof. (Theorem 3.3.2). □

When considering queries with more than one condition, one cannot classify the DAG G as being either in the special case or general case, but rather, must classify each condition Y_j into the cases. This is because a target X might be independent of $\alpha(Y_0)$ given a condition Y_0, but might still be dependent on $\alpha(Y_1)$ given Y_1. Then, Y_0 would be in the special case, whereas Y_1 would be in the general case. Resultingly, the query would not depend on Y_0 and its ancestors, but would still depend on Y_1 and its ancestors.

Furthermore, when considering multiple targets $X_0, X_1, ...X_i$, one must check whether a set of ancestors $\alpha(Y_j)$ is independent of all vertices $X_i, i \in [0, m]$ given Y_j. Otherwise, if one of the targets was dependent on an ancestor of Y_j, the query q would depend on that ancestor. From this, we arrive at Theorem 3.3.3.

Theorem 3.3.3. *Let $q = p(X_0, X_1, ..., X_m | Y_0, Y_1, ..., Y_n)$ be a query on a DAG G. Then the computation of q requires the following sets of vertices depending on the structure of G:*

1. *q always depends on X_i ,*
2. *q always depends on all vertices along a route from each vertex $X_i, i \in [0, m]$ to each vertex $Y_j, j \in [0, n]$, not including Y_j itself,*
3. *For each $Y_j, j \in [0, n]$ in the general case, that is, whenever all targets $X_i, i \in [0, m]$ are dependent on $\alpha(Y_j)$ given Y_j, q depends on vertices Y_j and $\alpha(Y_j)$.*

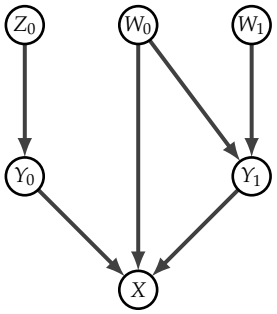
Therefore, if Y_j is in the special case, q does not depend on Y_j nor $\alpha(Y_j)$.

Figure 3.3.3: Graphical structure of the two cases: in the general case, X_is are not conditionally independent from all elements $Z_k \in \alpha(Y_j)$ given Y_j. In the special case, they are.

Example 3.3.2. Here, we apply Theorem 3.3.3 to determine the set of vertices a query q over the DAG G in Figure 3.3.4 depends on. An explicit calcuation to verify this is not given here; instead, we simply identify the set. Such calculations will take place in later examples.

Suppose $q = p(X|Y_0, Y_1)$. Then, for each of the conditions Y_0 and Y_1, we determine whether it is in the general case of the special case. Immediately, we know that the query depends on at least the target X.

- Y_0 has only one ancestor, namely Z_0. Z_0 is conditionally independent of X given Y_0. Therefore, the condition Y_0 is in the special case. Resultingly, q does not depend on Y_0 nor Z_0.
- Y_1 has two ancestors, namely W_0 and W_1. X is not conditionally independent from W_0 given Y_1, since we can find a route to W_0 which does not pass through Y_1. Therefore, Y_1 is in the general case, and we conclude that the query still additionally depends on W_0, W_1 and Y_1



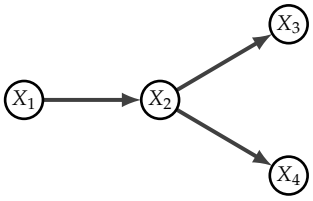
G

Figure 3.3.4

We conclude that $\Delta(G,q) = \{X, Y_1, W_0, W_1\}$.

The following examples verify which vertices are in the set $\Delta(G,q)$ outlined in Theorem 3.3.3. First we present several computations in the general case, then computations in the special case. Finally, we show how two different queries over the same DAG G can fall into each case.

Example 3.3.3. General Case. Let $q = p(X_3 = x_3, X_4 = x_4)$ be a query on $G = (V,E)$ in Figure 3.3.5. The computation of q is as follows.



G

Figure 3.3.5

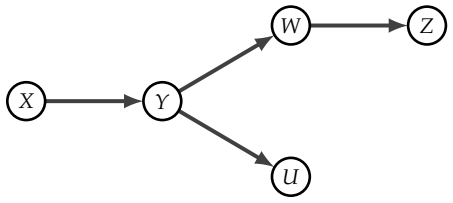
$$
\begin{aligned}
q &= p(X_3 = x_3, X_4 = x_4) \\
&= \sum_{x_2}\sum_{x_1} p(X_3 = x_3, X_4 = x_4 | X_2 = x_2) \cdot p(X_2 = x_2 | X_1 = x_1) \cdot p(X_1 = x_1) \quad (1) \\
&= \sum_{x_2}\sum_{x_1} p(X_3 = x_3 | X_2 = x_2) p(X_4 = x_4 | X_2 = x_2) \cdot p(X_2 = x_2 | X_1 = x_1) \cdot p(X_1 = x_1) \quad (2) \\
&= \sum_{x_2}\sum_{x_1} p(X_3 = x_3 | X_2 = x_2) \cdot p(X_4 = x_4 | X_2 = x_2) \cdot p(X_1 = x_1, X_2 = x_2) \quad (3)
\end{aligned}
$$

where equality (1) is by construction of the Bayesian network and equality (2) comes from the definition of conditional probability since X_3 and X_4 are conditionally independent. Equality (3) comes from the following application of conditional probability:

$$
p(X_2 = x_2 | X_1 = x_1) \cdot p(X_1 = x_1) \;=\; p(X_1 = x_1, X_2 = x_2).
$$

The query q depends on the targets X_4 and X_3 and their parents, X_2 and X_1.

Example 3.3.4. General case. The following example follows similar structure to Example 3.3.3 and uses real values for the conditional probabilities. Consider the graph $G = (V, E)$ from 3.3.6, where a vertex V takes a binary value $v \in [0, 1]$, and suppose we are given a query $q = p(Y = 1 | Z = 1)$.



G

Figure 3.3.6

Let the conditional probabilities be given by

$$
\begin{aligned}
&p(X=1)=0.2 \\
&p(Y=1|X=0)=0.6 \qquad\qquad p(Y=1|X=1)=0.4 \\
&p(W=1|Y=0)=0.4 \qquad\qquad p(W=1|Y=1)=0.3 \\
&p(Z=1|W=0)=0.7 \qquad\qquad p(Z=1|W=1)=0.3 \\
&p(U=1|Y=0)=0.8 \qquad\qquad p(U=1|Y=1)=0.4
\end{aligned}
$$

Then, the joint probability is given by

$$p(X,Y,W,U,Z) = p(X) \cdot p(Y|X) \cdot p(U|Y) \cdot p(W|Y) \cdot p(Z|W).$$

The aim of this example is to demonstrate that our query depends on the set of ancestors $\alpha(Y)$ and the nodes on route from Z to Y, in this case, the route $Y \to W \to Z$. For this computation, we set $Z=1$, and recall that since we are in a binary setting, for a vertex V we have $p(V=0) = 1 - p(V=1)$. Then, to determine $p(Y=1|Z=1)$, we first compute $p(Y=1)$:

$$
\begin{aligned}
p(Y=1) &= \sum_X p(Y=0|x) \cdot p(x) \\
&= 0.6 \cdot 0.8 + 0.4 \cdot 0.2 \\
&= 0.48 + 0.08 \\
&= 0.56 \\
P(Y=0) &= 1 - p(Y=1) = 0.44
\end{aligned}
$$

By similar calculation and plugging in the values $p(Y=1)$ and $p(Y=0)$, we obtain that $p(W=0) = \sum_Y p(W=0|Y)p(Y) \approx 0.65$ and $P(W=1) \approx 0.34$. Likewise, we compute $p(Z=1) \approx 0.56$ and $p(Z=0) \approx 0,43$.

From here, we determine how the observation $Z=1$ affects the likelihood of values of W and consequently Y. Bayes' theorem allows us to do the calcultion using quantities specified in the conditional probabilities:

$$p(W=1|Z=1) = \frac{p(Z=1|W=1) \cdot p(W=1)}{p(Z=1)} \approx \frac{0.3 \cdot 0.34}{0.56} \approx 0.18.$$

Subsequently $p(W=0|Z=1) \approx 0.81$. Now that we see how W is affected by the observation that $Z=1$, we can move upward in our route to determine who Y is affected by. Continuing with our condition that $Z=1$ and applying Bayes' theorem again,

$$p(Y=1|W=1) \;=\; \frac{p(W=1|Y=1)\cdot p(Y=1)}{p(W=1)} \;\approx\; \frac{0.18\cdot 0.44}{0.18} \;\approx\; 0.48.$$

Finally,

$$\begin{aligned} p(Y=1|Z=1) &= p(Y=1|W=0)\cdot p(W=0) + p(Y=1|W=1)\cdot p(W=1) \\ &\approx (0.59\cdot 0.81) + (0.48\cdot 0.18) \\ &\approx 0.17. \end{aligned}$$

Thus, by tracing the effects of the observation $Z=1$ on a route to $Y=1$ and using the ancestors $\alpha(Y)$, we have answered the query $q = p(Y=1|Z=1)$. Notice that this computation did not involve the vertex U, which is neither an ancestor of Y nor along the route between Y and Z.

G

Figure 3.3.7

Example 3.3.5. Special Case. Let $q = p(X_4 = x_4|X_2 = x_2)$ be a query on G in Figure 3.3.7. Then the query only involves vertices X_4, X_3, and X_2, so there is no need to consider vertex X_1. This is because X_4 is conditionally dependent on $\alpha(X_2) = X_1$.

$$p(X_4 = x_4|X_2 = x_2) = \sum_{x_3} p(X_4 = x_4|X_3 = x_3)\cdot p(X_3 = x_3|X_2 = x_2).$$

Example 3.3.6. This example compares two queries over the same graph G in Figure 3.3.8. Let $q_1 = p(X_2 = x_2|X_3 = x_3)$ and let $q_2 = p(X_3 = x_3|X_2 = x_2)$.

Note that X_2 is an ancestor of X_3, and trivially X_2 is not conditionally indepen-

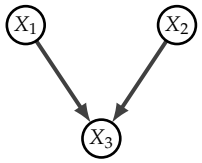


Figure 3.3.8

dent from X_2. Therefore q_1 is in the general case of Theorem 3.3.3, since the target of q_1 (namely X_2) is not disjoint from the ancestors of the condition, $\alpha(X_3)$.The computation of q_1 therefore requires the target X_2, the ancestors $\alpha(X_2)$, the condition X_3, and $\alpha(X_3)$. These sets together involve X_1, X_2, and X_3. This dependence is verified:

$$\begin{aligned} q_1 &= p(X_2 = x_2 | X_3 = x_3) \\ &= \frac{p(X_2 = x_2, X_3 = x_3)}{p(X_3 = x_3)} \\ &= \frac{\sum_{x_1} p(X_1 = x_1) \cdot p(X_2 = x_2) \cdot p(X_3 = x_3 | X_1 = x_1, X_2 = x_2)}{\sum_{x_1} \sum_{x_2} p(X_1 = x_1) \cdot p(X_2 = x_2) \cdot p(X_3 = x_3 | X_1 = x_1, X_2 = x_2).} \end{aligned}$$

Now we consider q_2. In this case, the conditional variable X_2 has no ancestors, and therefore we do not need to consider its distribution, since we are in the special case of Theorem 3.3.3. We show below that q_2 only relies only on X_3 and X_1.

Figure 3.3.9: X_2 effectively removed from the model by conditioning $X_2 = x_2$.

Intuitively, setting $X_2 = x_2$ effectively removed the variable from the model, since we are conditioning other variables on this observation. Once the other variables are adjusted, X_2 no longer plays a role since it already has an established value and does not have parents that must be considered. That is, we are setting $p(X_3 = x_3 | X_1 = x_1) = p(X_3 = x_3' | X_1 = x_1', X_2 = x_2)$ for all x_1', x_2'. Thus:

$$
\begin{aligned}
q_2 &= p(X_3 = x_3 | X_2 = x_2) \\
&= \frac{p(X_3 = x_3, X_2 = x_2)}{p(X_2 = x_2)} \\
&= p(X_3 = x_3) \\
&= \sum_{x_1} p(X_3 = x_3 | X_1 = x_1, X_2 = x_2) \cdot p(X_1 = x_1) \\
&= \sum_{x_1} p(X_3 = x_3 | X_1 = x_1) \cdot p(X_1 = x_1).
\end{aligned}
$$

Therefore, q_2 relies only on X_3 and X_1 as described in the special case, whereas q_2 relies on X_2 as well, as described in the general case.

Chapter 4

Markov Equivalence

4.1 Goal

In order to reduce the cost of a query on a graph G, we will exploit the observation that two Bayesian networks with distinguishable structures can model the same probability distribution. In this case, the corresponding graphs are called *Markov equivalent*. Given a query on a graph G, we search for a Markov equivalent graph G' such that the cost of the query is minimized when considered on G' instead of G.

To do so, we must develop a robust understanding of Markov equivalence. This includes: how to identify whether two graphs are Markov equivalent, how a graph can be altered while remaining in the set of Markov equivalent graphs (called the *Markov equivalence class*), how computations of queries on a graph are altered when the graph structure is altered, and how to search for minimizing graphs.

In this chapter, we present definitions, theorems, and examples to establish a sufficient understanding of Markov equivalence for this purpose.

4.2 Basic Properties of Markov Equivalence

Definition 4.2.1 (Markov equivalence [10]). *Two directed acyclic graphs G and G' are **Markov equivalent** if for every Bayesian network $B = (G, \theta_G)$, there exists a Bayesian network $B' = (G', \theta_{G'})$ such that B and B' describe the same probability distribution, and vice versa. This is denoted $G \sim_M G'$.*

Definition 4.2.2 (Markov equivalence class [10]). *The set of all DAGs which are Markov equivalent to a DAG G is called the* ***Markov equivalence class****(MEC) of G, denoted* $[G]$.

Significant research has been done to characterize Markov equivalence classes ([11], [10], [12]). This research has identified several key features of a graphs which allow us to more easily identify reversible edges and construct MECs. The following definitions outline several properties of graphs (and edges within them) which we will use to determine whether two graphs lie in the same MEC. Furthermore, it will help us build an understanding of how a graph can be manipulated without altering which MEC it belongs to.

Definition 4.2.3 (Covered Edge [11]). *Given a DAG* $G = (V, E)$*, an edge* $e = (X, Y) \in E$ *is called* ***covered*** *in G iff* $\prod_Y^G = \prod_X^G \cup X$*. That is,* (X, Y) *is covered in G iff X and Y have identical parents in G with the exception that X is not a parent of itself.*

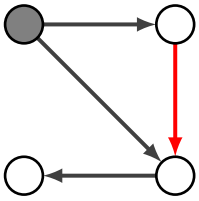
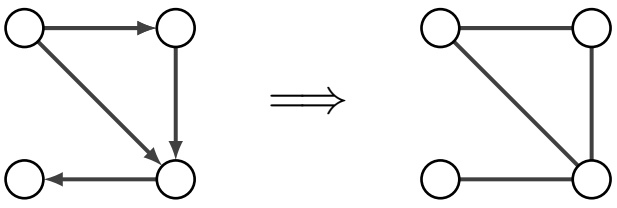


Figure 4.2.1

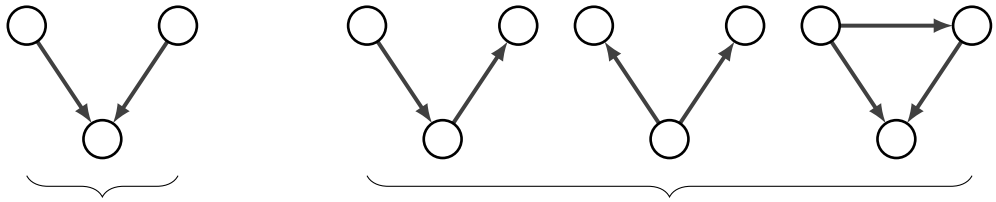
In Figure 4.2.1, for example, the red edge is the only covered edge, since the nodes it connects have a shared set of parents, namely, the single gray node.

Definition 4.2.4 (Skeleton). *Let* $G = (V, E)$ *be a directed graph. The undirected graph* $G' = (V, E')$ *which is constructed by removing the orientation of all edges in E is called the* ***skeleton*** *of G.*



G $\qquad\qquad$ $G' = \text{skeleton}(G)$

Definition 4.2.5 (v-structure). *A set of three vertices X, Y, Z of a graph $G = (V, E)$ is called a **v-structure** (or immorality) iff $(X, Y) \in E$ and $(Z, Y) \in E$ but X and Z are not adjacent.*



v-structure *not v-structures*

Definition 4.2.6 (Irreversible edge [12]). *Let G be a DAG. A directed edge $(X, Y) \in G$ is called **irreversible** in G if changing (X, Y) to (Y, X) either creates or destroys a v-structure, or creates a directed cycle. An edge which is not irreversible is called **reversible**.*

Lemma 4.2.1 (Criterion for Markov equivalence [10]). *Two graphs are equivalent iff they have the same skeleton and v-structures.*

Proof. A sketch of the proof is provided, as the complete proof involves many definitions and details which are not presented in this paper. First, we use the definition of an **active route** (sometimes called active path) with respect to a set $\mathbb{Z}$: a route $[X_0, X_1, ...X_n]$ in which all nodes X_i are themselves **active**, meaning that (1) every middle node of a v-structure with respect to $\mathcal{Z}$ either is or has a descendant in $\mathcal{Z}$, and (2) every other node in the route is outside $\mathcal{Z}$.

The proof depends on two lemmas presented in the same paper: **Lemma 1** has the consequence that adjacency of nodes is invariant among equivalent DAGs. **Lemma 2** has the consequence that one can determine the presence or absence of v-structures by observing certain seperation properties of the graph alone, and vice versa. The two lemmas are unified with an inductive step to show that active paths in one DAG correspond to active paths in a Markov equivalent DAG. Furthermore, properties of active paths allow us to conclude that the DAGs must have the same skeleton and v-structures. □

Lemma 4.2.2 (Covered edges are exactly reversible edges [11]). *Let $G = (V, E)$ be a DAG, let $X, Y \in V$ and let $e = (X, Y) \in E$. Let $G' = (V, E')$ be the DAG constructed from G by reversing the edge (X, Y) to (Y, X). Then G and G' are equivalent iff e is a covered edge in G.*

Proof. Let $G = (V, E)$ be a DAG. A sketch of the proof is as follows (see [11] for the full proof):

(**if**) Suppose $e = (X, Y)$ is a covered edge in G. Let G' be equivalent to G, except that e is reveresed to $e' = (Y, X)$. Then:

- G' is also a DAG. To show this, suppose that G' is not a DAG. Since G and G' only differ by $e' = (Y, X)$, there must be a directed cycle incuding e' in G'. Then there is a path in G from Y to X. By assumption, Y and X have a shared set of parents. However, one can conclude from this that G also contains a cycle because every cycle mut include at least one node from $\prod_x^G$, and therefore is not a DAG, a contradiction. By this contradiction, we conclude that G' must be a DAG.
- $G' \sim_M G$. Firstly, G and G' trivially have the same skeleton, since they only differ by the orientation of edge e. Then, if they were *not* Markov equivalent graphs, they would differ by a v-structure. If G' has a v-structure not present in G, it must include the edge (Y, X). However, this would imply that X has a parent which is not a neighbor to Y, contradicting the assumption that e is covered. A similar argument holds for the scenario where G has a v-structre not present in G'.

(**only if**) We now show that if $e = (X, Y)$ is not a covered edge in G, we are in one of two cases: either G' contains a directed cycle, or G' is a DAG which is not equivalent to G. If (X, Y) is not a covered edge in G, then at least one of the following hold:

1. There is a node $Z \neq X$ which is a parent of Y but not of X.
2. There s a node W which is a parent of X but not of Y.

Let $Z \neq X$ be a parent of Y in G but not a parent of X in G. If Z and X are not neighbors, then there is a v-structure consisting of edges (X, Y) and (Y, Z) in G which does not exist in G'. If X is a parent of Z in G, then it must also be a parents of Z in G', which would imply that G' contains a directed cycle and is therefore not a DAG.

The argument for the second case is equivalent, assuming W is a parent of X in G and deriving a v-structure with $(W, X) and (X, Y)$ which exists in G' but not G. We conclude that G would have to contain a directed cycle. In both scenarios, we have derived a contradiction which shows that (X, Y) must be covered in G.

□

Example 4.2.1 (Members of Markov equivalence class [10]). Consider the four DAGs in Figure 4.2.2.

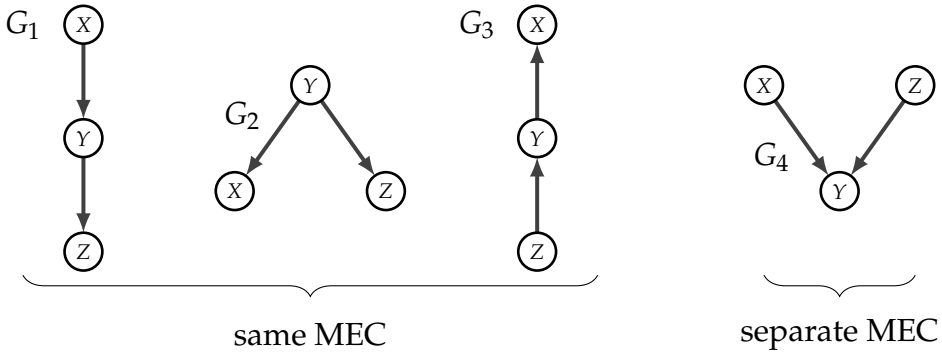


Figure 4.2.2

The fact that G_1, G_2, and G_3 are in the same MEC while G_4 is not can be seen by considering the joint probability distribution $p(X, Y, Z)$ of each of the graphs.

$$\begin{aligned} G_1: &\quad p(X,Y,Z) = p(X) \cdot p(Y|X) \cdot p(Z|Y) \\ G_2: &\quad p(X,Y,Z) = p(Y) \cdot p(X|Y) \cdot p(Z|Y) \\ G_3: &\quad p(X,Y,Z) = p(Z) \cdot p(Y|Z) \cdot p(X|Y) \end{aligned}$$

By Bayes' theorem, all of the values of G_2 can be completely determined from values from G_1:

$$p(X)p(Y|X) = p(X,Y) = p(Y)p(X|Y)$$

and $p(Z|Y)$ is unchanged. A similar application of Bayes' theorem shows that G_3 is completely determined by G_1 and G_2:

$$p(z) \cdot p(Y|Z) = p(Z,Y) = p(Y) \cdot p(Z|Y)$$

where $p(Y)$ can be attained directly from G_2 and $p(Z|Y)$ can be attained from G_1. Therefore, we can conclude that since all three describe the same distribution, they lie in the same equivalence class. In contrast, the joint probability for G_4 is given by

$$p(X,Y,Z) = p(X) \cdot p(Z) \cdot p(Y|X,Z)$$

which can not be determined from the values of G_1, G_2 and G_3. This is because in $[G_1]$ X is conditionally independent from Y and Z, that is, X is only indepen-

dent in the circumstance that we are given values for Y and Z. Meanwhile, in G_4, X is marginally independent of Z, that is, independent in the circumstance that we ignore Y. Conditional independence does not provide information about marginal independence, and conversely, marginal independence does not imply conditional independence. Therefore, G_4 describes a different distribution, and does not lie in $[G_1]$.

Corollary 1. *Lemma 4.2.1 and Lemma 4.2.2 together imply that the graph G' which results from only reversing the orientation of a covered edge $(X, Y) \in G$ is in the MEC of $[G]$.*

Proof. This follows from the fact that edge reversal does not change the skeleton of the graph G (since no edges are added nor removed) and reversing covered edges does not alter the v-structures of G (Lemma 4.2.2), thus the v-structures and skeleton are unchanged. □

4.3 Effects of Edge Reversal

Lemma 4.3.1. *Given a single directed edge between two parentless nodes in which the joint probability $p(X_1, X_2) = p(X_1) \cdot p(X_2|X_1)$, the new factorization of the joint probability distribution $p(X_1, X_2)$ can be computed as follows using only the known quantities $p(X_1)$ and $p(X_2|X_1)$:*

$$p(X_1, X_2) \;=\; \sum_{X_1} p(X_2|X_1) \cdot p(X_1) \frac{p(X_2|X_1) \cdot p(X_1)}{\sum_{X_1} p(X_2|X_1) \cdot p(X_1)}.$$

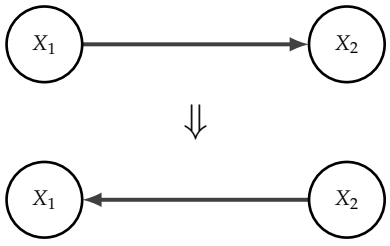


Figure 4.3.1

Proof. This follows directly from Bayes' theorem, which gives us

$$p(X_1|X_2) = \frac{p(X_2|X_1) \cdot p(X_1)}{p(X_2)}$$

$$= \frac{p(X_2|X_1) \cdot p(X_1)}{\sum_{X_1} p(X_2|X_1) \cdot p(X_1)}.$$

□

Lemma 4.3.2. *Given a directed edge between two nodes X_1 and X_2 with a shared parent node X_P, the joint probability $p(X_1, X_2, X_p)$ after switching edge (X_1, X_2) to (X_2, X_1) can be computed as follows using only known quantities $p(X_p)$, $p(X_1|X_p)$, and $p(X_2|X_1, X_p)$:*

$$p(X_1, X_2, X_P) = [\frac{p(X_2|X_1, X_p) \cdot p(X_1|X_p)}{\sum_{X_1} p(X_1|X_p) \cdot p(X_2|X_1, X_p)}] \cdot [\sum_{X_1} p(X_1|X_p) \cdot p(X_2|X_1, X_p)] \cdot p(X_p).$$

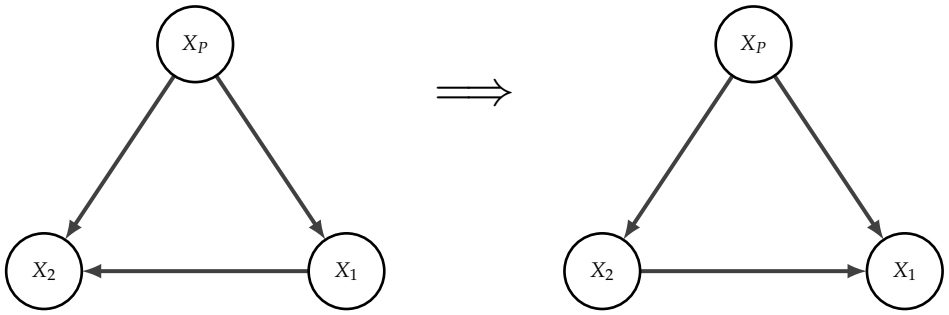


Figure 4.3.2

Remark. Furthermore, because this result does not depend on $p(X_p)$, we can conclude that for a set of shared parents $P = \{X_{P_1}, X_{P_2}, \ldots X_{P_i}\}$ of two nodes X_1 and X_2, the same equality holds. Additionally, note that any reversal of an edge with a set of shared parents is MEC-invariant, since all such edges are covered (corollary 1).

Proof. From before the edge reversal, we have access to the values $p(X_p), p(X_1|X_p)$, and $p(X_2|X_1, X_p)$. Our goal is to determine the new joint probability distribution $p(X_1, X_2, X_p)$ using this information. Therefore, we must find $p(X_p), p(X_2|X_p)$, and $p(X_1|X_2, X_p)$. First, $p(X_p)$ is already trivially available. Then, to compute $p(X_1|X_2, X_p)$,

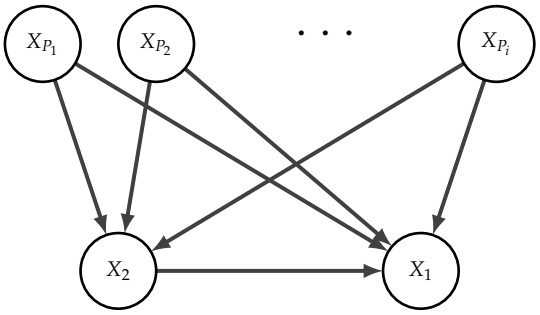


Figure 4.3.3

$$\begin{aligned} p(X_1|X_2,X_p) &= \frac{p(X_1,X_2,X_p)}{p(X_2,X_p)} \\ &= \frac{p(X_2|X_1,X_p)\cdot p(X_1,X_p)}{p(X_2,X_p)} \\ &= \frac{p(X_2|X_1,X_p)\cdot p(X_1|X_p)\cdot p(X_p)}{p(X_2|X_1,X_p)} \\ &= \frac{p(X_2|X_1,X_p)\cdot p(X_1|X_p)}{p(X_2|X_p)} \end{aligned}$$

which depends only on known quantities once we compute

$$\begin{aligned} p(X_2|X_p) &= \frac{p(X_2,X_p)}{p(X_p)} \\ &= \frac{1}{p(X_p)}\sum_{X_1} p(X_1,X_2,X_p) \\ &= \frac{1}{p(X_p)}\sum_{X_1} p(X_1,X_p)\cdot\frac{p(X_2,X_1,X_p)}{p(X_1,X_p)} \\ &= \sum_{X_1} p(X_1|X_p)\cdot p(X_2|X_1,X_p). \end{aligned}$$

Together, these give us

$$\begin{aligned} p(X_1, X_2, X_p) &= p(X_1|X_2, X_p) \cdot p(X_2|X_p) \cdot p(X_p) \\ &= [\frac{p(X_2|X_1, X_p) \cdot p(X_1|X_p)}{\sum_{X_1} p(X_1|X_p) \cdot p(X_2|X_1, X_p)}] \cdot [\sum_{X_1} p(X_1|X_p) \cdot p(X_2|X_1, X_p)] \cdot p(X_p). \end{aligned}$$

□

Remark. Notice that the above cancels to show that the two graphs are Markov equivalent, since $p(X_1|X_2, X_p) \cdot p(X_2|X_p) \cdot p(X_p) = p(X_2|X_1, X_p) \cdot p(X_1|X_p) \cdot p(X_p)$.

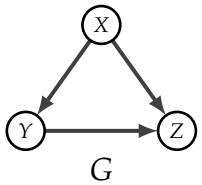
4.4 Identifying Reversible Edges

Definition 4.4.1 (Essential edge [12], [14]). *An **essential edge** (also called a compelled edge [11]) of a graph $G = (V, E)$ is an edge $(X, Y) = e \in E$ such that for every graph $G' = (V, E')$ in $[G]$, the orientation of $e \in E$ is unchanged. That is, there is no graph $G' = (V, E')$ in $[G]$ such that $(Y, X) \in E'$.*

Definition 4.4.2 (Essential graph [12]). *The **essential graph** of a Markov equivalence class $[G]$ is a mixed graph G_* such that the only directed edges in G_* are essential edges of $[G]$. That is,*

- *The directed edge $e = (X, Y) \in G_*$ iff $e \in G$ for every graph $G \in [G]$,*
- *The undirected edge $X, Y \in G_*$ iff there are graphs G_1 and $G_2 \in [G]$ such that (X, Y) in G_1 and (Y, X) in G_2.*

Example 4.4.1. In this example, we identify the essential edges of a Markov equivalence class and construct the essential graph. Consider the following DAG $G = (V, E)$:



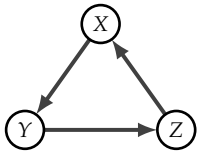
G

Figure 4.4.1

We use the following realizations to show that the graph has no essential edges.

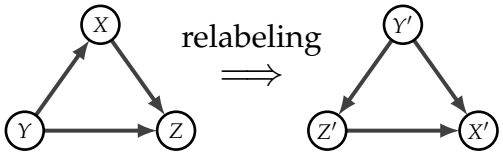
If we switch any edge in G, the resulting graph G' becomes one of two cases:

- G' is cyclic, and therefore no longer a DAG, meaning this edge was irreversible.
  Example: reversing edge (X, Z) as in **(A)** creates a cycle.



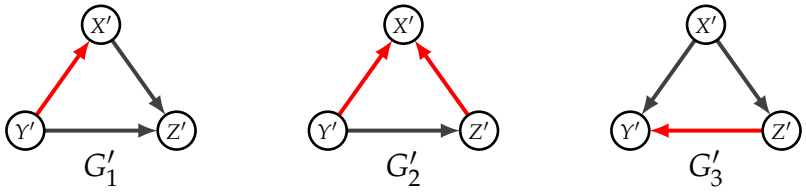
(A)

- G' is equivalent to G up to relabeling, and therefore in the same MEC.
  Example: reversing (X, Y) and changing labels according to $X \to Y'$, $Y \to Z'$, and $Z \to X'$, as in **(B)**.

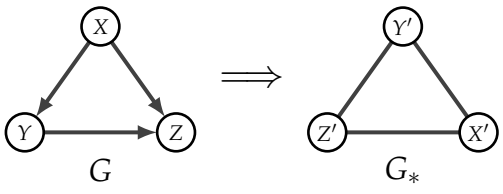


(B)

Thus the following graphs are included in $[G]$, though they are not the entire MEC. The red edges in each are those with orientation which differs from edges in G. Notice, however, that some relabeling was required, meaning these edges do not necessarily correspond (under this new labeling) to covered edges.



Therefore, under some circumstance, for every edge $(U, V) = e \in G$, there is a DAG $G' \in [G]$ such that $(V, U) \in G'$. This means that G has no essential edges. We construct the essential graph G_* of G by removing the orientation of all non-essential edges in G:



Remark. Andersson et. al. [12] demonstrate that essential graphs are consistent across a MEC and unique to it, meaning an essential graph can be used as a representative for a given MEC. Furthermore, essential graphs are not necessarily DAGs, and therefore do not generally belong to the MEC that they represent.

This can be seen by the fact that the distinction between two MECs lies entirely in either differences in their skeletons or in the orientations of their compelled edges.

"It is only in the heart that one can see rightly; what is essential is invisible to the eye." -Antoine de Saint-Exupéry, The Little Prince.

Remark. One should take care to note the distinction between implications about non-essential edges versus covered edges:

- For $G = (V, E)$, an edge $e \in E$ is non-essential iff there exists a graph G' in $[G]$ such that e is reversed,
- An edge e in G is covered iff the graph G' derived from *only* reversing e, is in the MEC of G. (Follows from corollary 1.)

That is, reversing e *exclusively* while remaining in the MEC of G requires e to be covered. Meanwhile, a non-essential edge $e \in G$ may require other edges in G to be reversed as well to remain in the same MEC. This means that not all non-essential edges can be reversed at a given time. The following example demonstrates that some configurations of non-essential edge reversals are not possible if we wish to remain in the MEC of G.

Example 4.4.2. Again consider members of the Markov equivalence class seen in example 4.2.1. The positions of the nodes are visually rearranged for clarity, but no properties are changed.
Notice that edge $(Y, Z) \in G_1$ is non-essential, since there exists another graph (namely G_3) in $[G_1]$ such that $(Z, Y) \in G_3$. Therefore, we know that there exists some combination of edge reversals such that (Y, Z) can be reversed without leaving $[G_1]$. However, if we choose to switch only (Y, Z) to (Z, Y), we have exactly graph G_4, which is no longer in the same MEC. Therefore, the fact that an edge is non-essential does not provide information about the circumstances in which the edge can be reversed without altering the MEC; only that in some context, it can be. This realization leads us to corollary 2.

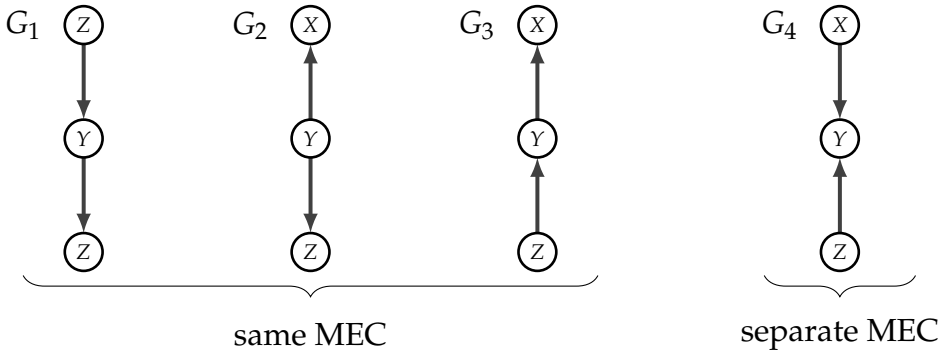


Figure 4.4.2

Corollary 2. *Let E_c be the set of covered edges in G, and let E_e be the set of essential edges in G. Then $E_c \subseteq \bar{E}_e$, the set of non-essential edges in G.*

Proof. This follows directly from the realization that non-essential edges can be switched while remaining in the same MEC under the correct conditions, while covered edges can be switched immediately, without regard to other edge conditions (the conditions for their reversal are always met). □

Example 4.4.3. We use the graph G from example 4.4.1 to demonstrate corollary 2. Of course, this is not sufficient for a proof, but rather helps to contextualize the lemma.

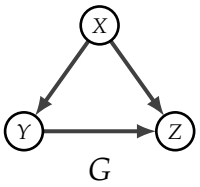


Figure 4.4.3

In example 4.4.1 we determined that all edges $e_i \in E$ of $G = (V, E)$ are non-essential, so the set of non-essential edges $E_G^N = \{(X,Y),(Y,Z),(X,Z)\}$. We can quickly see that the only covered edge of G is (Y,Z) since Y and Z share a single parent X, not including Z itself. Then, indeed, $\{(Y,Z)\} \subset E_G^N$.

Remark. There are two scenarios we consider for finding reversible edges, that is, edges which can be reversed without altering the MEC of the graph:

- Firstly, we may wish to find the set of edges which can be immediately

reversed in one step, without regard to other edges in the graph. This coincides with the set of covered edges.

- Secondly, we may wish to find the set of edges which can ever be reversed within a graph, under the correct conditions of other edges being reversed beforehand/simultaneously. This coincides with the set of non-essential edges.

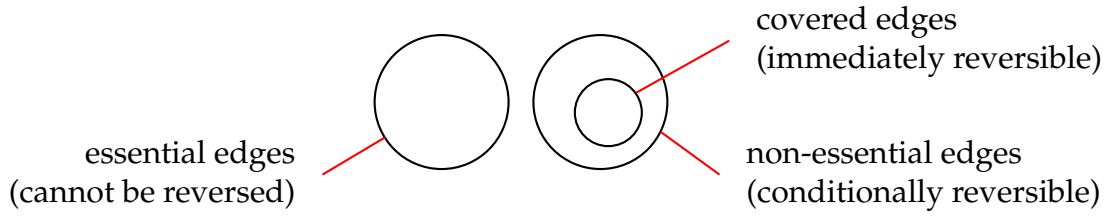


Figure 4.4.4

Theorem 4.4.1 (Edge reversal sequence. [11]). *Given two Markov equivalent DAGs G and G', there exists a sequence of distinct edge reversals in G with the following properties:*

- *Each edge reversed in G is a covered edge,*
- *After each reversal, G is a DAG and $G \sim_M G'$*
- *After all reversals, $G = G'$.*

That is, we can transform G into G' via a finite sequence of covered edge reversals, such that all of the intermittently resulting graphs $G_1, G_2, ...G_n \sim_M G, G'$.

Proof. Chickering's [11] proof defines the procedure **Procedure Find-Edge** which takes a DAG G as input. At each step, the procedure identifies a next edge to reverse. To be reversible, such an edge must be covered, by Lemma 4.2.2. The proof demonstrates that such an edge is identifiable when G and G' remain unequal, and each reversal reduces the number of edges in total which must be reversed, eventually transforming G into G'. □

Chapter 5

Exploiting Markov Equivalence using Greedy Strategy

5.1 Motivation

One complication of considering Markov equivalence is that Markov equivalence classes can be quite large. None of the graphs in the MEC is necessarily a more natural representation of the probabilistic dependencies than the others.

We therefore begin by exploring a *greedy* strategy. The idea behind this strategy takes advantage of Theorem 4.4.1:

> *Given two arbitrary graphs G and G' in a family of Markov equivalent graphs, G can be transformed into G' through a finite sequence of transformations such that 1) each step in the sequence involves reversing a single directed edge, and 2) after each edge reversal, the resulting graph is contained within the same Markov equivalence class.*

This result allows us to develop the greedy strategy, an algorithm in which each step entails searching for a single edge that can be reversed without leaving the original Markov equivalence class, and simultaneously reduces the number of variables which must be sampled to answer our original query.

There are two aspects we wish to consider while searching for a reversible edge which reduces the cost of answering a query:

- Does reversing the edge alter existing v-structures in the graph, either by creating a new v-structure or destroying an existing one?

- Does switching the edge indeed benefit our query, ie, reduce the number of variables that must be considered while computing the probabilities?

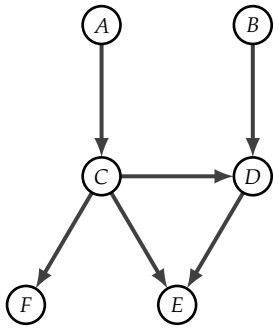
In this section, we present some relevant background, utilize a procedure from [11] to identify reversible edges, develop a framework for when edge switching is beneficial, and analyze the efficacy of the procedure for minimization.

5.2 Greedy strategy background

The [11] proof of Theorem 4.4.1 uses the algorithm **Procedure Find Edge** which returns and edge that can be reversed at each step. We will exploit this same algorithm to find reversible edges for our minimization, and proceed to evaluate whether the reversal is beneficial. In order to undertand the procedure, we must first make use of the following definition.

Definition 5.2.1. *Let $G = (V, E)$ be a DAG. A topological sort on G is a linear ordering of V such that for every edge $(X, Y) \in E$, vertex X comes before vertex Y in the ordering. The node with the smallest value with respect to the sort is called **minimal**. Likewise, the highest valued node is called **maximal**.*

Example 5.2.1. Topological sorts are not necessarily unique. To see this, consider the graph $G = (V, E)$ from Figure 5.2.1.



G

Figure 5.2.1

One can quickly verify that all of the following assignments of values to vertices are valid topological sorts. This is not an exhaustive list, but demonstrates that multiple orderings are possible.

A	B	C	D	E	F
1	2	3	4	6	5
2	1	4	3	5	6
2	1	5	3	5	6

Lemma 5.2.1. *All DAGS have at least one valid topological ordering.*

Proof. Let G be a DAG. Then, G contains no directed cycles. We use the following claim to construct the proof:

Claim: All DAGs have at least one vertex with no incoming edges. This can be shown by condtradiction: Suppose that all vertices in a DAG G have an incoming edge. Then, for an arbitrary vertex X, there is at least one incoming edge from a vertex Y. Similarly, Y has at least one incoming edge, say from a vertex Z. One can continue upward by this reasoning, but since DAGs are finite, will eventually need to repeat a node. Repeating a node, however, tells us that there exists a cycle, which implies G is not a DAG. Therefore, the claim holds.

Having verified this claim, one can construct an algorithm to assign a valid topological sort to any DAG G.

1. Identify a vertex X with no incoming edges, assign it with a value x.
2. Construct a DAG G' by removing X (and edges it participates in) from G. G is a DAG, since no edges have been added, and therefore no cycles.
3. Repeat from step 1 using G', assigning X to the value $x + 1$.

The above algorithm is always well defined, all iterations are over a DAG, and by construction ensures that for every edge (X, Y) in G, X appears before Y in the ordering and is not repeated. □

Lemma 5.2.2. *The edge (X, Y) output from Algorithm 1 is a covered edge.*

Proof. (Contradiction). Suppose the edge (X, Y) from Algorithm 1 with inputs (G, G') is not a covered edge. Then, without loss of generality, Y has a parent $Z \neq X$ which is not a parent of X. Then we have two cases:

1. If Z is not adjacent to X, then (X, Y) participates in a v-structure, namely edges (X, Y) and (Y, Z). Then, G has a v-structure which cannot be present

Algorithm 1: PROCEDURE FIND-EDGE returns a covered edge in a DAG

Input: Two equivalent DAGs $G = (V, E)$ and $G'(V', E')$ that differ by at least one edge

Output: An edge which differs between G and G'

1 Perform a topological sort on the nodes in G.

2 Let Y be the minimal node with respect to the topological sort for which $\prod_Y^G \neq \emptyset$.

3 Let X be the maximal node with respect to the sort for which $X \in \prod_Y^P$.

return (X, Y)

in G'. However, this contradicts $G \approx_M G'$, since they must have the same v-structures. Therefore, in this case, we have a contradiction and (X, Y) must therefore be covered.

2. If Z is adjacent to X, then either (X, Z) or (Z, X) is an edge in G. If (X, Z) in G, then either (X, Z) or (Z, Y) must differ between G and G', otherwise G' would have to have directed cycle (X, Y, Z). If (X, Z) differs between G and G', then the same arguments apply using Z in Step 2 of the algorithm rather than Y. Likewise if (X, Y) differs between G and G', then one can choose Z instead of X in Step 3. Therefore, we once again derive a contradiction, and conclude that (X, Y) must be covered.

□

5.3 Implementation

5.4 Cost

Chapter 6

Comparison of methods

A comparison of the speed of normal querying VS the speed of the new query + time required to exploit ME.
Ultimately trying to answer the questions:

- Can we do faster inference?
- Under what circumstances is it possible/effective?
- How much faster is it?
- Maybe include a section on future work in this chapter.

Bibliography

[1] S. Greenland, J. Pearl, and J. Robins, "Causal diagrams for epidemiological research," *Epidemiology 10*, pp. 37–48, 1999.

[2] M. Glymour and S. Greenland, "Causal diagrams," *Modern Epidemiology, 3rd edition*, pp. 183–209, 2008. ed. Lippincott Williams & Wilkins, Philidelphia, PA.

[3] P. L. Miller and P. R. Fisher, "Causal models for medical artificial intelligence," *Selected Topics in Medical Artifical Intelligence*, 1988. ISBN: 978-1-4613-8779-4.

[4] P. Szolovits and S. G. Pauker, "Categorical and probabilistic reasoning in medical diagnosis," *Artificial Intelligence, Volume 11, Issues 1-2*, pp. 115–144, 1978.

[5] M. Shwe, B. Middleton, D. Heckerman, M. Henrion, E. Horvitz, H. Lehmann, and G. Cooper, "A probabilistic reformulation of the quick medical reference system," *Annual Symposium on Computer Application in Medical Care*, pp. 790–794, 1990.

[6] S. Morgan and C. Winship, "Counterfactuals and causal inference: Methods and principles for social research (analytical methods for social research)," 2007. Cambridge University Press, New York, N.

[7] H. Blalock, Jr., "Causal models in the social sciences," *The Economic Journal Vol. 82 No. 328.*, pp. 1420–1423, 1972. Published by Oxford University Press.

[8] D. Cox and N. Wermuth, "Causality: a statistical view," *International Statistical Review 72*, pp. 285–305, 2004.

[9] S. Lauritzen, "Causal inference from graphical models," *Complex Stochastic Systems*, pp. 63–107, 2001. Chapman and Hall/CRC Press, Boca Raton, FL.

[10] T. Verma and J. Pearl, "Equivalence and synthesis of causal models," *Proceeding of the Sixth Annual Conference on Uncertainty in Artificial Intelligence, UAI '90*, pp. 255–270, 1990. DOI: 10.1142/S0218488504002576.

[11] D. M. Chickering, "A transformational characterization of equivalent bayesian network structures," *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence UAI'95*, 1995. Morgan Kaufmann Publishers Inc.

[12] S. Andersson, D. Madigan, and M. Perlman, "A characterization of markov equivalence classes for acyclic digraphs," *Annals of Statistics 25*, 1997.

[13] J. Pearl, "Causal inference in statistics: an overview," *Statistics Surveys Vol. 3*, pp. 96–146, 2009.

[14] I. Flesch and P. Lucas, "Markov equivalence in bayesian-network structures," *J. Mach. Learn. Res., 2*, 2002.

[15] R. D. Schachter, "Evaluating influence diagrams," *Operations Research Vol. 34, No. 6 (Nov. - Dec. pp. 871-882)*, 1986.

[16] T. A. Stephenson, "An introduction to bayesian network theory and usage," *Dalle Molle Institute for Perceptual Artificial Intelligence*, 2000. IDIAP-RR 00-03.

[17] C. M. Grinstead and J. L. Snell, "Introduction to probability," *American Mathematical Society*, 2003. Copyright (C) 2006 Peter G. Doyle.

Declaration of Authorship

I hereby declare that I have completed this work independently and using only the sources and tools specified. The author has no objection to making the present Master's thesis available for public use in the university archive.

Ich erkläre, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe. Seitens des Verfassers bestehen keine Einwände die vorliegende Masterarbeit für die öffentliche Benutzung im Universitätsarchiv zur Verfügung zu stellen.

Ort, Abgabedatum
(Place, date)

Unterschrift der Verfasserin
(Signature of the Author)