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, March 10, 2020

Abstract

Zusammenfassung

Acknowledgements

Notation and Abbreviations

- **DAG** Directed Acyclic Graph
- **iff** If and only if
-

Contents

Chapter 1

Introduction

1.1 Motivation

Many fields rely on prediction and classification of new information using known data, processes which require encoding statistical relationships into analyzable structures. Conditional dependencies between random variables in probability distributions can be effectively modeled using Bayesian networks, giving us a reliable way to do inference on our model.

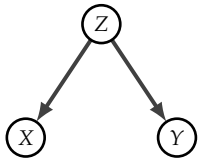
For instance, in the medical field, we may have information about a large number of patients (such as height, age, sex, blood type, etc.) and we may wish to use this information to predict whether a patient has a specific disease or not. In this scenario, we would need to model the known data and then query the model for the probability that, given these factors, a patient has the disease.

Naturally, it is in our best interest to create a model and a method for querying which allow us to do inference on our probabilistic models as efficiently as possible, in order to achieve results more quickly. This thesis explores a method for answering queries more efficiently in the context of graphical models.

1.2 Background

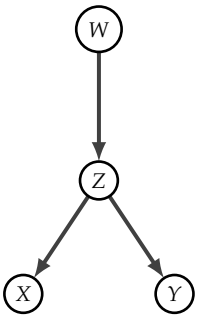
The family of models explored in this thesis are directed graphical models over discrete probability distributions. The purpose of graphical models is to encode information about vectors of random variables, namely the interdependencies satisfied by the vector's joint probability function. For example, 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 following graph:



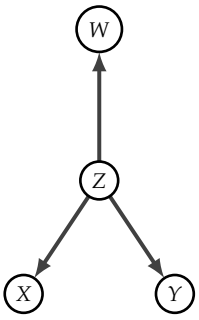
$$p(X,Y|Z) = p(X|Z)p(Y|Z)$$

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 described by a variety of different graph structures which ultimately describe the same underlying probability distribution. This will be explored in more detail later. To understand how the complexity of answering a query depends on graph structure, consider the following two models:



(A)

$$p(X|Z)p(Y|Z)p(Z|W)p(W)$$



(B)

$$p(X|Z)p(Y|Z)p(W|Z)p(Z)$$

If we wish to answer a query about the random variable Y in graph **(A)**, for instance, we must also compute the probabilities of each vertex it depends on either directly or indirectly, that is, all of its **ancestors** (in this case Z and W in addition to Y itself). Alternatively, if we wish to answer a query about Y in graph **(B)**, we need only to compute Z and Y. The number of variables which must be sampled is reduced when we consider **(B)** instead of **(A)**.

In a context where answering queries about a single vertex may require us to sample arbitrarily many other vertices in the graph, we seek to find a graphical representation of our probability distribution which minimizes the number of vertices that must be sampled when answering a query.

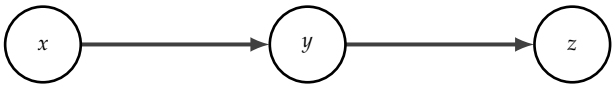
1.3 Goal

As mentioned previously, there are certain circumstances in which two Bayesian networks with different graphical structures can describe the same underlying probability distribution. That is, querying the graphs will produce statistically indistinguishable results. Such sets of graphs are called *Markov equivalent*.

The goal of this research is as follows: given a query on a graph G (which represents a single factorization of the probability distribution we wish to query), find a Markov equivalent graph G' which minimizes the number of variables that must be computed to answer the query. The purpose of the minimization is to reduce the cost of querying in order to achieve faster inference, as well as evaluate the gained speed-up versus the cost of transforming the graph structure.

1.4 Motivating Example

The following example demonstrates how answering a query can be sped up by querying a Markov equivalent graph with a different structure. Consider the following DAG:



(A)

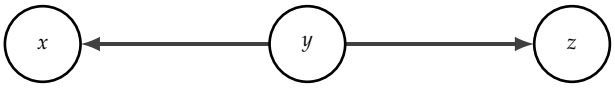
with conditional probabilitis given by

$$p(x=1) = 0.2$$
$$p(y=1|x=0) = 0.3 \qquad p(y=1|x=1) = 0.4$$
$$p(z=1|y=0) = 0.1 \qquad p(z=1|y=1) = 0.9$$

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 following DAG:



(B)

Here, we compute the conditional probabilities of **(B)** to show that the model can equivalently describe our joint probability distribution from **(A)**, 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

$$p(x|y) = \frac{p(y|z) \cdot p(x)}{p(y)} = \frac{p(y|x) \cdot p(x)}{\sum_x p(y|x) \cdot p(x)}$$

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 **(B)** is an equivalent model to **(A)**. If we

answer the query $p(z|y)$ on model **(B)**, 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. Verma and Pearl[1] 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. Flesch and Lucas [2] 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 Bayesian Statistics. Chickering [3] explores parameters of equivalent networks, presents a procedure for altering graphs without changing the described distribution, and quantifies the complexity of several such manipulations. Andersson et. al. [4] make similar headway by exploring reductions of graphs to representative form, as well as describing procedures by which one can alter the structure.

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

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.*

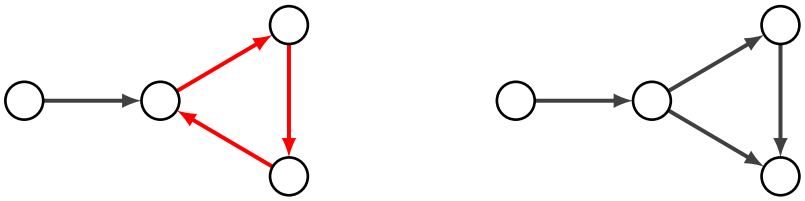
*G is called **undirected** if every edge in G is undirected, meaning that for each edge $(u, v) \in E$, the edge (v, u) is also in E, for $u \neq v$. Otherwise, G is called **directed**. We denote such undirected edges by $\{u, v\}$ or equivalently $\{v, u\}$. A **mixed graph** (also called a hybrid graph) is a graph G which contains both directed and undirected edges.*

Definition 2.1.2 (Route). *A **route** in a graph $G = (V, E)$ is a sequence $v_1, v_2, ..., v_k$ of vertices in V where there is an edge connecting v_i to v_{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 2.1.3 (Path). *Path*

Definition 2.1.4 (Directed cycle). *Directed cycle*

Definition 2.1.5. *A 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 (abbreviated DAG for the remainder of this paper).*



not a DAG DAG

The directed cycle is shown above in red.

Remark. Every undirected graph is a chain graph.

Definition 2.1.6 (Parent). *Given two vertices $u, v \in V$ in a directed graph $G = (V, E)$, u is called a **parent** of v if the edge $(u, v) \in E$. The set of parents of a node v in a graph $G = (V, E)$ is denoted $\prod_v^G$.*

Definition 2.1.7 (Neighbor). *Given two vertices $u, v \in V$ in an undirected graph $G = (V, E)$, u is called a **neighbor** of v if $\{u, v\} \in E$.*

2.2 Foundations of Bayesian Statistics

Definition 2.2.1 (Bayesian Network). *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 X.*

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

Chapter 3

Inference without Markov Equivalence

- Details of the problem statement
- Querying method without exploiting MEC (Depth first or breadth first search of dependent variables, ie ancestors)
- Speed of this querying method.

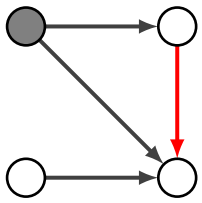
Chapter 4

Markov Equivalence

Definition 4.0.1 (Markov equivalence [1]). *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 $G \sim_M G'$.*

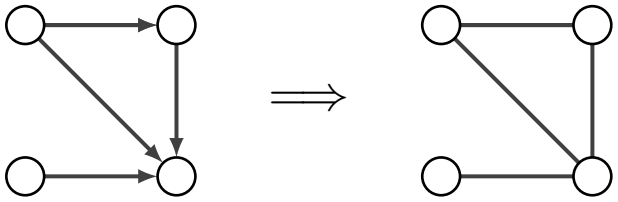
Definition 4.0.2 (Markov equivalence class [1]). *The set of all DAGs which are Markov equivalent to a DAG G is called the **Markov equivalence class**(MEC) of G, denoted $[G]$.*

Definition 4.0.3 (Covered Edge [3]). *Given a DAG $G = (V, E)$, an edge $e = (u, v) \in E$ is called **covered** in G iff $\prod_u^G = \prod_v^G \cup v$. That is, (u, v) is covered in G iff u and v have identical parents in G, with the exception that u is not a parent of itself.*



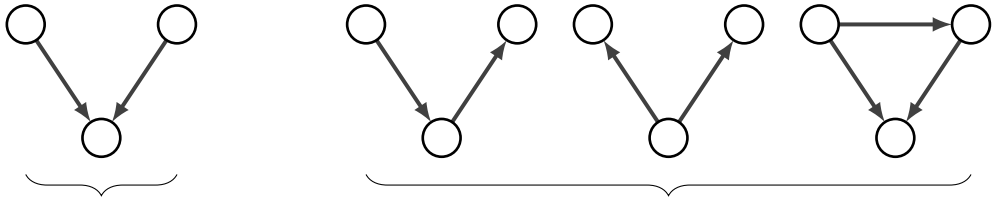
Here, for example, the red edge is the only covered edge, since the nodes it connects have a shared set of parents, namely, the single gray node.

Definition 4.0.4 (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 4.0.5 (v-structure). *A set of three vertices u,v,w of a graph $G = (V,E)$ is called a **v-structure** (or immorality) iff $(u,v) \in E$ and $(w,v) \in E$ but u and w are not adjacent.*



v-structure $\qquad\qquad$ *not v-structures*

Definition 4.0.6 (Irreversible edge [4]). *Let G be a DAG. A directed edge $(u,v) \in G$ is called **irreversible** in G if changing (u,v) to (v,u) either creates or destroys a v-structure, or creates a directed cycle. An edge which is not irreversible is called **reversible**.*

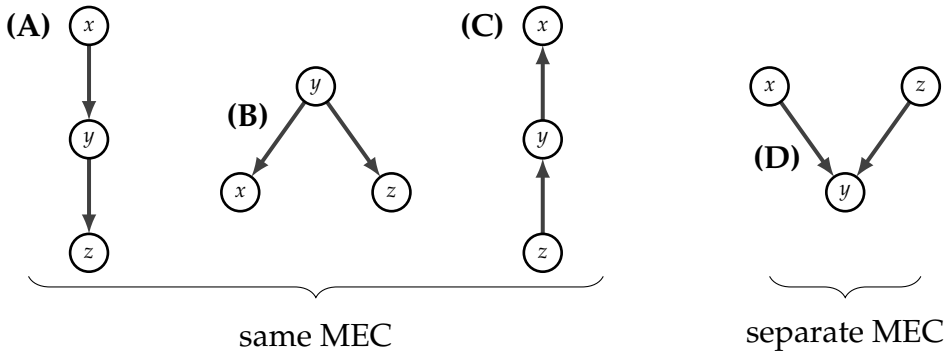
Lemma 4.0.1 (Covered edges are exactly reversible edges [3]). *Let $G = (V,E)$ be a DAG, let $u,v \in V$ and let $e = (u,v) \in E$. Let $G' = (V,E')$ be the DAG constructed from G by reversing the edge (u,v) to (v,u). Then G and G' are equivalent iff e is a covered edge in G.*

Lemma 4.0.2 (Criterion for MEC-equivalence [1]). *Two graphs are equivalent iff they have the same skeleton and v-structures.*

Corollary 1. *Lemma 4.0.1 and Lemma 4.0.2 together $\implies$ the graph G' which results from only reversing the orientation of a covered edge $(u,v) \in G$ 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 4.0.1), thus the v-structures and skeleton are unchanged. □

Example 4.0.1 (Members of Markov equivalence class [1]). Consider the following four DAGs:



The fact that **(A), (B)**, and **(C)** are in the same MEC while **(D)** is not can be seen by considering the joint probability distribution $p(x,y,z)$ of each of the above graphs.

(A) $p(x,y,z) = p(x) \cdot p(y|x) \cdot p(z|y)$
(B) $p(x,y,z) = p(y) \cdot p(x|y) \cdot p(z|y)$
(C) $p(x,y,z) = p(z) \cdot p(y|z) \cdot p(x|y)$

By Bayes' theorem, all of the values of **(B)** can be completely determined from values from **(A)**:

$$p(x)p(y|x) = p(x,y) = p(y)p(x|y)$$

and $p(z|y)$ is unchanged. A similar application of Bayes' theorem shows that **(C)** is completely determined by **(A)** and **(B)**:

$$p(z) \cdot p(y|z) = p(z,y) = p(y) \cdot p(z|y)$$

where $p(y)$ can be attained directly from **(B)** and $p(z|y)$ can be attained from **(A)**. Therefore, we can conclude that since all three describe the same distribution, they lie in the same equivalence class. Alternatively, the joint probability for **(D)** is given by

$$p(x,y,z) = p(x) \cdot p(z) \cdot p(y|x,z)$$

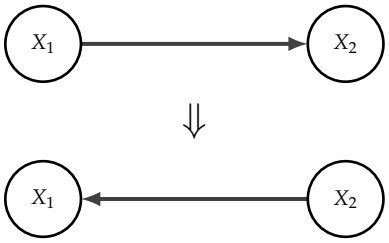
which can not be determined from the values of **(A), (B)** and **(C)**. This is because in **[(A)]**, x is conditionally independent from y and z, that is, x is only independent in the circumstance that we are given values for y and z. Meanwhile, in **(D)**, x is marginally independent of z, that is, 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, **(D)** describes a different distribution, and does not lie in **[(A)]**.

4.1 Effects of Edge Reversal

Lemma 4.1.1. *Given a single directed edge between two parentless nodes in which* $p(X_2) = p(X_1) \cdot p(X_2|X_1)$*, the joint probability after switching edge orientation is given by*

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



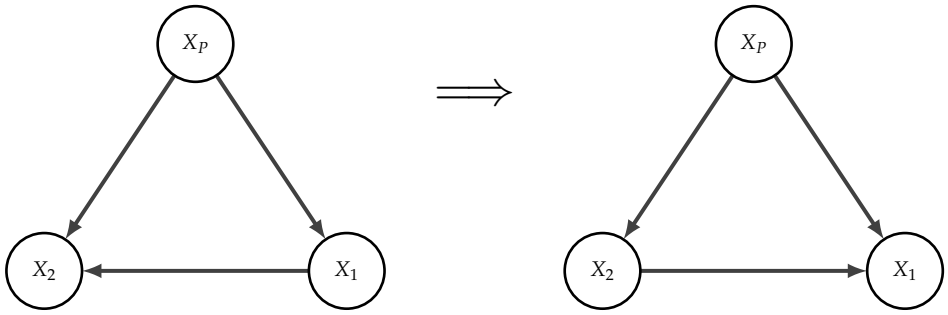
Proof. This follows directly from Bayes' theorem, which gives us

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

□

Lemma 4.1.2. *Given a directed edge between two nodes* X_1 *and* X_2 *with a shared parent node* X_P*, the joint probability after switching edge* (X_1, X_2) *to* (X_2, X_1) *is given by*

$$p(X_1, X_2, X_P) \;=\; p(X_P)\cdot p(X_2|X_P)\cdot p(X_1|X_2, X_P) \;=\; \ldots$$



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 $p(X_1|X_1, X_p)$,

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

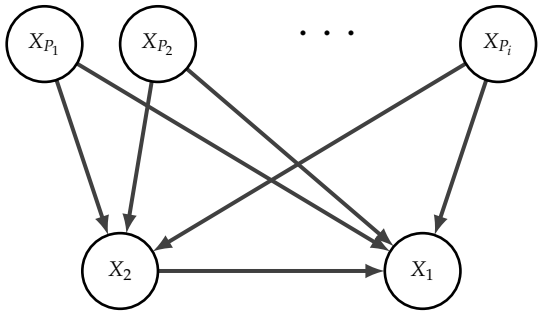
which depends only on known quantities once we compute

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

TO BE CONT.

□

Remark. Note that the result of Lemma 4.1.2 does not depend on $p(X_p)$. We can therefore 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, this equality holds. Furthermore, note that any reversal of an edge with a set of shared parents is MEC-invariant, since all such edges are covered (corollary 1).



4.2 Find Reversible Edges of a DAG Model

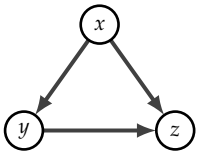
Definition 4.2.1 (Essential edge [4] [2]). *An **essential edge** (also called a compelled edge [3]) of a graph $G = (V, E)$ is an edge $(u, v) = e \in E$ such that for every graph $G' = (V, E')$ in $[G]$, the orientation of $e \in E$ is unchanged. That is, there is no graph $G' = (V, E')$ in $[G]$ such that $(v, u) \in E'$.*

Definition 4.2.2 (Essential graph [4]). *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* $e = (u,v) \in G_*$ *iff* $e \in G$ *for every graph* $G \in [G]$*,*
- *The undirected edge* $u^-v \in G_*$ *iff there are graphs* G_1 *and* $G_2 \in [G]$ *such that* (u,v) *in* G_1 *and* (v,u) *in* G_2*.*

Example 4.2.1. In this example, we identify the essential edges of a Markov equivalence class and construct the essential graph. Consider the following DAG $G = (V,E)$:

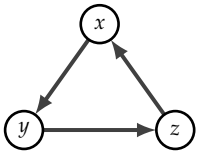


G

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.

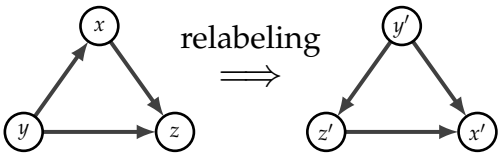
  Example: reversing edge (x,z) as in **(A)** creates a cycle.



(A)

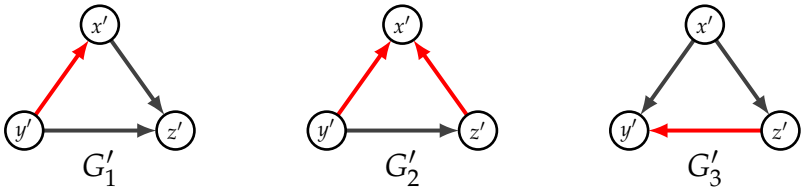
- G' is equivalent to G up to relabeling, and therefore in the same MEC.

  Example: reversing (x,y) and changing labels according to $x \to y'$, $y \to z'$, and $z \to x'$, as in **(B)**.

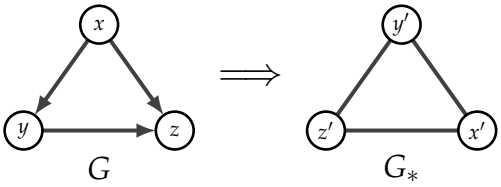


(B)

So 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. [4] 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.

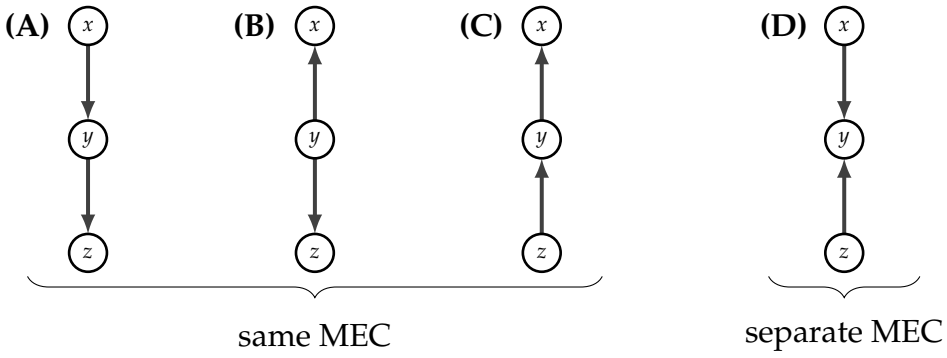
"It is only in the heart that one can see rightly; what is essential is invisible to the eye."
-Antoine de Saint-Exupéry, The Little Prince.

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 4.2.2. Again consider members of the Markov equivalence class seen in example 4.0.1. The positions of the nodes are visually rearranged for clarity, but no properties are changed.



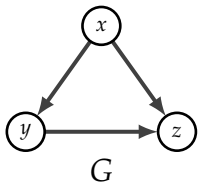
Notice that edge $(y,z) \in$ **(A)** is non-essential, since there exists another graph (namely **(C)**) in **[(A)]** such that $(z,y) \in$ **(C)**. Therefore, we know that there exists some combination of edge reversals such that (y,z) can be reversed without leaving **[(A)]**. However, if we choose to switch only (y,z) to (z,y), we have exactly graph **(D)**, 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.

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 4.2.3. We use the graph G from example 4.2.1 to demonstrate corollary 2. Of course, this is not sufficient for a proof, but only helps us contextualize

the lemma.



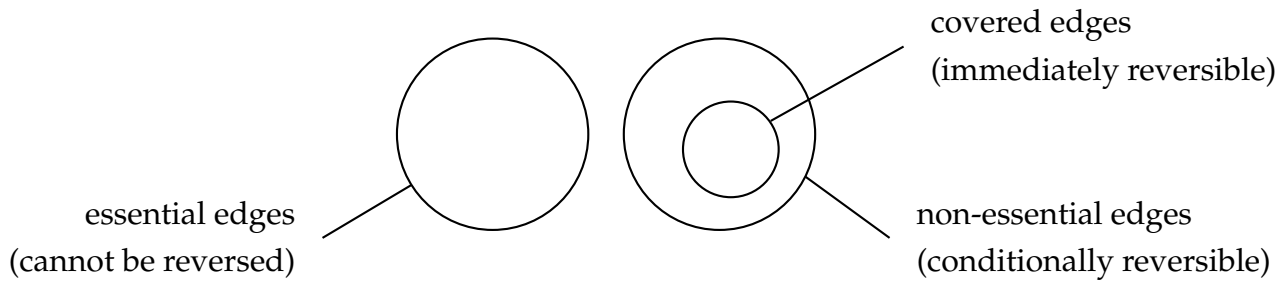
G

In example 4.2.1 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.

Algorithm 1 responds to the first scenario.



Theorem 4.2.1 (Edge reversal sequence. [3]). *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,*

Algorithm 1: COVERED EDGES returns a set of covered edges in a DAG

Input: A directed acyclic graph G
Output: The set of covered edges in G

```
covered_edges = []
for e = (x, y) ∈ Edges(G) do
    for p_i ∈ ∏_x^G, p_i ≠ y do
        if p_i ∏_y^G then
            pass
        else
            add e to covered_edges
    end
end
return covered_edges
```

- *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 $G_1, G_2, ...G_n \sim_M G, G'$.

4.3 Essential Graph of a DAG Model

We quickly see that construction of the essential graph G_* of a DAG model G is quite complicated, as it involves identifying all essential edges of G, which in turn involves having access to or a characterization of all DAGs in the MEC $[G]$. Andersson et. al. [4] have done significant work to characterize all DAGs which lie within a given MEC, and have presented a polynomial time algorithm (The Construction Algorithm) to construct the unique essential graph associated with a MEC. The algorithm does not require and exhaustive search over the entire MEC.

A quick overview of the arguments made by Andersson et al. is as follows:

4.4 Random interesting things

Theorem 4.4.1 (Parents of equivalent graphs [3]). *Let G and G' be any pair of Markov equivalent DAGs. If G has a node with exactly k parents, then G' has a node*

Algorithm 2: THE CONSTRUCTION ALGORITHM returns the essential graph of a DAG

Input: A directed acyclic graph $G = (V, E)$
Output: The essential graph G_* of G

1 Define $G_0 := G$.

2 For $i \geq 1$, convert every directed edge $(x, y) \in G_{i-1}$ that is not strongly protected in G_{i-1} into an undirected edge x^-y, obtaining a graph G_i. Stop after k steps, where $k \geq 0$ is the smallest nonnegative integer such that $G_k = G_{k+1}$. Necessarily, $k \leq |E|$

3 This algorithm produces a sequence $G \sim_M G_0 \subset ... \subset G_k = G_{k+1}$.

with exactly k parents.

Chapter 5

Exploiting Markov Equivalence using Greedy Strategy

5.1 Motivation

One complication of considering Markov equivalence is that a Markov equivalence class 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 4.2.1:

> *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 an edge which can be switched to minimize 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 benefit our query, ie, reduce the number of variables that must be considered while computing the probabilities?

The former informs whether the switch is possible, that is, whether we remain in the same MEC after switching the edge, since graphs in the same MEC have the same v-structures. The latter considers whether the switch is useful.
In the following algorithm, we begin by considering the number of parents of variables of interest, with the following explanation:

- If the random variable X has 0 parents, then there is no incoming edge to X_0 that we can switch.
- If X has two or more parents, then there is an existing v-structure, so no edge $(X_{0,i}, X_0)$ can be reversed without altering the MEC.
- If X has one parent, there is an opportunity for edge reversal, so we move to the next part of the algorithm.

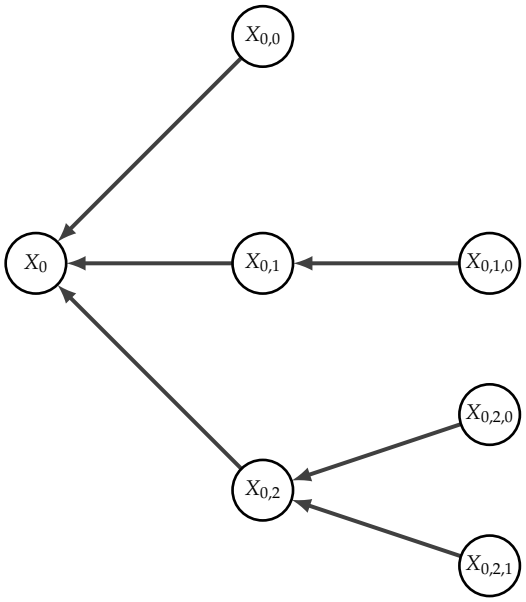
Algorithm 3: REVERSED EDGE returns a DAG with a beneficial edge reversal for query Q

Input: A directed acyclic graph G
Output: G' (= G with beneficially switched edge (X_u, X_v))

```
X = X_0 if #∏^G_{X_0} = 1 then
    continue
else
    for X_{0,i} ∈ ∏^G_{X_0}, where i = 0,1, ... #∏^G_{X_0} do
        X = X_{0,i}
    end
return G'
```

5.2 Examples

Example 5.2.1. Consider the following graph G, and suppose we are given the query $p(X_0)$, where $p(X_0) = p(X_0|X_{0,0}, X_{0,1}, X_{0,2})$ and $p(X_{0,1}) = p(X_{0,1}|X_{0,1,0})$ and $p(X_{0,2}) = p(X_{0,2}|X_{0,2,0}, X_{0,2,1})$: RETURN TO THIS



G

The returned graph G' is equivalent to G except that edge $(X_{0,1,0}, X_{0,1})$ in G has been reversed in G'. Notice that computing $p(x)$ in G requires computing $p(X_0|X_{0,1})$ and $p(X_{0,1}|X_{0,1,0})$, which considers three nodes. In G', $p(X_0)$ only requires us to consider nodes X_0 and $X_{0,1}$ since $X_{0,1}$ no longer depends on $X_{0,1,0}$. This is given by $p(X_0) = p(X_0|X_{0,1})$. We have therefore found an MEC equivalent graph to G which reduces the number of variables which must be computed from 3 to 2.

Two followup questions:

- How do we know which edges to return reversed if there are multiple possible reversible edges? Does that ever happen? Clearly yes.
- This clearly only considers immediately reversible edges, not non-essential edges as well. Is there a way to search for non-essential edges effectively?
- What is the best way to refer to my *target* node in a query, and all nodes upstream? Also, I should add some details about which nodes we consider before the algorithm explanation, and why those limit us to advanagous switches by default
- What does this look like in a non-tree structure? SHORT answer it seems to work perfectly: give another example (the one you wrote out) asap.

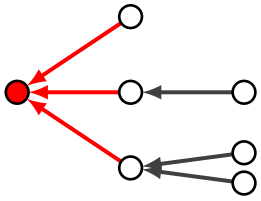
Set $X = X_0$

Check number of parents of X_0:

$\#\prod_{X_0}^{G} = 3 \neq 1$

Set $X = X_{0,0}$

no reversible edge; continue.

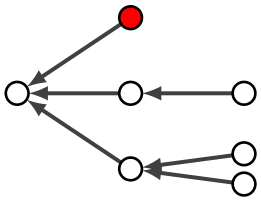


Check number of parents of $X_{0,0}$:

$\#\prod_{X_{0,0}}^{G} = 3 \neq 1$

Set $X = X_{0,1}$

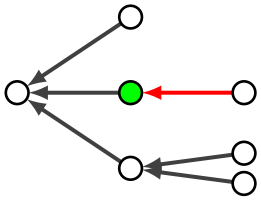
no reversible edges; continue.



Check number of parents of $X_{0,1}$

$\#\prod_{X_{0,1}}^{G} = 3 \neq 1$

possibly reversible edge.

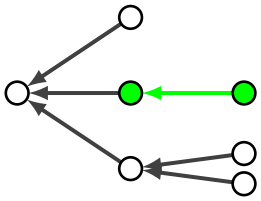


Check number of parents of $X_{0,1,0}$

$\#\prod_{X_{0,1,0}}^{G} = 0$

edge $(X_{0,1,0}, X_{0,0})$ is reversible.

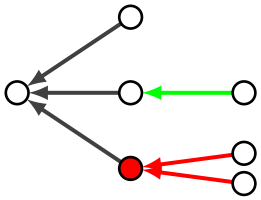
Set $X = X_{0,2}$



Check number of parents of $X_{0,2}$

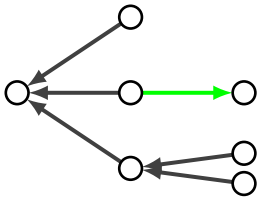
$\#\prod_{X_{0,2}}^{G} = 2 \neq 1$

no new reversible edges.

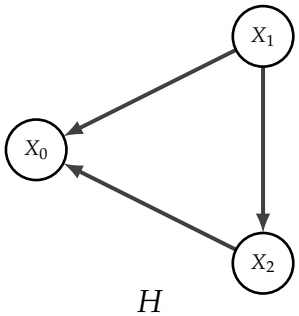


Reverse edge $(X_{0,1,0}, X_{0,0})$

Return graph G'.



Example 5.2.2. The following example demonstrates that the algorithm is effective outside of tree-structures. Consider the three node graph H below:



H

Here one can quickly see, for example, that if there had been an additional node X_3 and edge (X_3, X_0), the algorithm would return an unaltered graph, as desired, since there would no longer be reversible edges.

5.3 Implementation

5.4 Cost

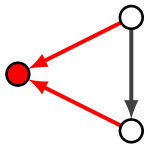
Set $X = X_0$

Check number of parents of X_0:

$\#\prod_{X_0}^{G} = 2 \neq 1$

Set $X = X_1$

no reversible edge; continue.

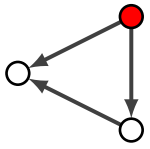


Check number of parents of X_1:

$\#\prod_{X_1}^{G} = 0 \neq 1$

Set $X = X_2$

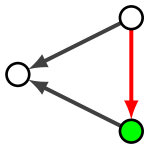
no reversible edges; continue.



Check number of parents of X_2

$\#\prod_{X_2}^{G} = 1$

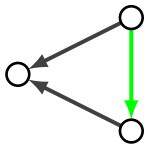
possibly reversible edge.



Check number of parents of X_1

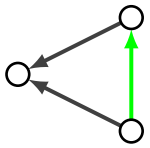
$\#\prod_{X_{0,1,0}}^{G} = 0$

edge (X_1, X_2) is reversible.



Reverse edge (X_1, X_2)

Return graph H'.



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] T. Verma and J. Pearl, "Equivalence and synthesis of causal models," *Proceeding of the Sixth Annual Conference on Unverstainty in Artificial Intelligence, UAI '90*, 2004. DOI: 10.1142/S0218488504002576.

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

[3] 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.

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

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)