UNIVERSITY OF COPENHAGEN
DEPARTMENT OF MATHEMATICAL SCIENCES

Master thesis

Tobias Ellegaard Larsen

Score-based causal discovery in a setting with tiered background knowledge

Date: May 31st 2024

Supervisors: Anne Helby Petersen and Claus Thorn Ekstrøm, University of Copenhagen
Internal supervisor: Niels Richard Hansen, University of Copenhagen

Abstract

Causal discovery with tiered background knowledge leads to more informative estimations of the true underlying data-generating DAG. In this thesis, we develop two score-based causal discovery algorithms that incorporate tiered background knowledge: Temporal Greedy Equivalence Search (TGES) and Simple Temporal Greedy Equivalence Search (Simple TGES). To be able to define the algorithms, we introduce needed graph theory, relevant score criteria, tiered background knowledge, and Greedy Equivalence Search (GES). We prove that TGES will always estimate a tiered MPDAG and that Simple TGES is sound and complete in the limit of large sample size. The algorithms have been implemented in R. We perform a simulation study, here amongst using a new metric on the performance of causal discovery algorithms. The simulation study shows that TGES outperforms Simple TGES and GES with respect to both adjacencies and orientations of edges. The computational time of TGES is slower than GES but shows potential to become faster. We finish the thesis by discussing the findings and methods used throughout.

Table of content

1 Introduction

The main concern of this thesis is within the field of causal discovery, also referred to as causal learning. In causal discovery, we assume that data is generated with some underlying causal structure, and we then try to estimate this structure. We will represent the structure of causal effects between variables by a graph with nodes corresponding to variables and directional edges corresponding to causal effects. We will say there is a causal effect of A on B if a change in A results in a change in B. We will, throughout the thesis, have the assumption commonly referred to as *causal sufficiency*. Causal sufficiency assumes there are no unobserved confounders or selection variables.

In causal discovery, we distinguish between constraint-based and score-based algorithms (Peters et al., 2017). Constraint-based algorithms seek to predict the underlying data-generating graph using conditional independence tests. The most popular constraint-based algorithm is the PC algorithm (named after Peter Spirtes and Clark Glymour, Spirtes et al., 1993). In this thesis, the focus is not constraint-based algorithms; hence, we refer to Section 7.2.1 in Peters et al. (2017) for further information on this. Instead, we will focus on the theory and application of score-based causal discovery methods and use constraint-based methods only for comparison. In this thesis, we will emphasize on the score-based methods on data with structural advantages that can assist in discovering possible causal effects between variables.

Score-based causal discovery algorithms score the graphs on their ability to fit the data. In practice, it is not feasible to score every possible graph, so we use an algorithm called Greedy Equivalence Search (GES). GES finds a local optimum by adding and removing edges until the score can no longer be improved. In this thesis, we will see results that show that in the limit of large sample size, this is sufficient for being sound and complete.

Life-course studies will often have variables that have been observed in different stages of a person's life, and we can rule out causal effects going from a later stage in life to an earlier stage. This sort of induced information is, in the field of causal discovery, referred to as tiered background knowledge. The use of tiered background knowledge extends to any type of data that contains information on the sequential structure but is highly relevant for data in a temporal setting. Earlier work in the field of causal discovery with tiered background knowledge is, for example, Petersen et al. (2021), which introduced the constraint-based algorithm Temporal PC (TPC).
We see from TPC that tiered background knowledge can better the estimate and we wonder whether there should be a score-based alternative to TPC. When investigating the supplementary material of J. D. Ramsey and Andrews (2017), we discover how the predictions of GES and PC differ from

each other. We see that when focusing on edge prediction without taking direction into account, GES tends to estimate a bigger proportion of the true graph, especially for denser graphs. We also see that GES does a better job than PC at directing the edges of the graph. Thus motivating extending GES into incorporating information from temporal ordering similarly to TPC as it might inherit the same desirable performance attributes as GES.

While no earlier work had dealt with the exact issue of tiered background knowledge with a greedy score-based causal discovery algorithm, this presented the opportunity to do exploratory research into how such a method could be made. We have developed and implemented two main methods, Simple Temporal GES (Simple TGES)(Algorithm 2) and Temporal GES (TGES)(Algorithm 3). Simple TGES is a post-hoc method that utilizes tiered background knowledge after estimating the graph by the use of data. We will show that Simple TGES can be proven to be sound and complete in the limit of large sample size (Theorem 2). TGES is a method that restricts which edges to score and how to score them according to the tiered background knowledge. In this way, TGES ensures every step of the way that the tiered background knowledge is accounted for, but the question of sound and completeness of TGES will, in this thesis, remain conjectures (Conjectures 1 and 2). TGES utilizes a new scoring criterion called Temporal Bayesian Information Criterion (TBIC) (Definition 8) that takes tiered background knowledge into account. We showcase properties of this new scoring criterion (Propositions 2 to 6). To test the performance of these new methods, we conducted a simulation study. We have implemented a new metric that allows for the comparison of causal discovery methods that utilize tiered background knowledge and general causal discovery methods. We have implemented everything in R, and the code can be found in my GitHub repository (Larsen, 2024).

First, in Section 1.1, we will present the notation and graph terminology used throughout this thesis. Next in Section 2 we will introduce the score-based discovery algorithm Greedy Equivalence Search made by D. M. Chickering (2003), and show that the algorithm is both sound and complete. Following this, we will introduce causal discovery in a temporal setting in Section 3, here we will introduce background knowledge in Section 3.1, in Section 3.3 define and present a new scoring criterion and in Sections 3.2 and 3.4 we will introduce two new score-based causal discovery algorithms that utilize information induced by a temporal setting. Afterwards, in Section 4, we bring forward the theory and implementation needed for conducting a simulation study on the performance of the newly introduced algorithms. In Section 5 we showcase findings from this simulation study. In the last section, we conclude this thesis with a discussion of the new theory presented, the results of the simulation study, and further research questions motivated by this thesis.

1.1 Graph terminology

We assume familiarity with the concept of *Directed Acyclic Graph* (DAG), *d-separation* and the *Markov property* also known as the causal Markov assumption (Hernan & Robins, 2020). We will define a graph $\mathcal{G}$ as a set of *nodes* V and *edges* $\mathbf{E}$, such that $\mathcal{G} = (V, \mathbf{E})$. We define another type of graph called *Partially Directed Acyclic Graph* (PDAG) that allows for undirected edges in $\mathbf{E}$ while still having acyclicity.

We will consider a data set consisting of d random variables $\mathbf{X} = (X_1, ..., X_d)$, and an observed sample size of m i.i.d. observations being realizations of those variables, such that the data set is $\mathbf{D} = (\mathbf{X_1}, ..., \mathbf{X_m})$. In this way, $\mathbf{D}$ is a $d \times m$ matrix of observations.
We will in this thesis assume that the variables are realizations of a multivariate Gaussian distribution with a mean dependency structure according to a DAG and parameters θ. We will refer to this DAG as the true data-generating DAG or the underlying data-generating DAG. We will denote the distribution of the underlying data-generating DAG $\mathcal{G}$ as p and the density as $p(\mathbf{D}|\theta, \mathcal{G})$.

Throughout this thesis, we represent the dependency structure of the data by the use of graphs. Each variable is represented by a single node in the graph. We will in some instances present a graph $\mathcal{G} = (V, \mathbf{E})$ by the use of an *adjacency matrix*. An adjacency matrix A_m is a $d \times d$ matrix with entries $a_{i,j} = 1$ if and only if there is an edge in $\mathbf{E}$ from node X_i to node X_j where $X_i, X_j \in V$. An undirected edge between X_i and X_j is presented in the adjacency matrix as $a_{i,j} = a_{j,i} = 1$. For example, we would have that the adjacency matrix of the DAG presented in Figure 1 would be

$$A_m = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \end{array} \begin{array}{c} \begin{array}{ccccccc} A & B & C & D & E & F & G \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

We may present the underlying data-generating DAG using a *weighted adjacency matrix*. A weighted adjacency matrix W_m is a $d \times d$ matrix with entries $w_{i,j}$ where $i, j \in \{1, ..., d\}$ and $w_{i,j}$ is the weight of causal effect of the of node X_i on node X_j. Each node X_i can be written as all nodes that have a causal effect on it, times the weight of the causal effect of nodes, plus an exogenous multivariate

Gaussian distribution ϵ_i:

$$X_i = \begin{pmatrix} X_1 & \dots & X_d \end{pmatrix} \begin{pmatrix} w_{1,j} \\ \vdots \\ w_{d,j} \end{pmatrix} + \epsilon_i$$

We have that $w_{i,j} = 0$ if and only if there is no causal effect i.e. no directed edge from node X_i to node X_j.

We define the *skeleton* of a graph as the graph obtained by changing all directed edges to undirected edges.

We say that two nodes are *adjacent* or share an *adjacency* if there is an edge from one of the nodes to the other node.

A *v-structure* is three nodes X_1, X_2 and X_3 where $X_1 \to X_2 \leftarrow X_3$ where X_1 and X_3 are non-adjacent. In Figure 1 this would be $E \to G \leftarrow F$.

A *Markov equivalence class* is the class which consists of all DAGs that have the same conditional independencies, we will denote this equivalence class $\mathcal{E}$ and often just refer to it as an equivalence class. We denote the equivalence class of a graph $\mathcal{G}$ as $\mathcal{E}(\mathcal{G})$.

A *Completed Partially Directed Acyclic Graph* (CPDAG) can represent a Markov equivalence class (Peters et al., 2017). A CPDAG has a directed edge wherever all DAGs in the Markov equivalence class have the same edge orientation and the CPDAG has an undirected edge wherever there is not complete agreement in all DAGs in the Markov equivalence class. The completeness refers to the CPDAG fully representing all conditional independence statements in any DAG in the corresponding Markov equivalence class through its d-separation statements. Whenever there is an undirected edge in the CPDAG, the orientation of the edge cannot be determined from the probability distribution of the variables. A CPDAG is uniquely identified by its v-structures and its skeleton. In Figure 2, we have an example of a CPDAG where the DAG in Figure 1 is contained in the Markov equivalence class represented by the CPDAG. As we can see, Figures 1 and 2 have the same skeleton and v-structures.

A *subgraph* of a graph $\mathcal{G} = (V, \mathbf{E})$, consists of a subset of the nodes $X_i \in V$ and all edges in $\mathbf{E}$ between these nodes.

A *parent* of a node X_i is any node that has a directed edge going into X_i. The set of parents of X_i in $\mathcal{G}$ is denoted $Pa_i^{\mathcal{G}}$.

A *descendant* of a node X_i is any node that has a path of directed edges from X_i going into it. The set of descendants of X_i in $\mathcal{G}$ is denoted $De_i^{\mathcal{G}}$.

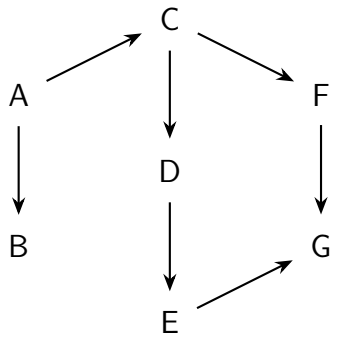


Figure 1: DAG example

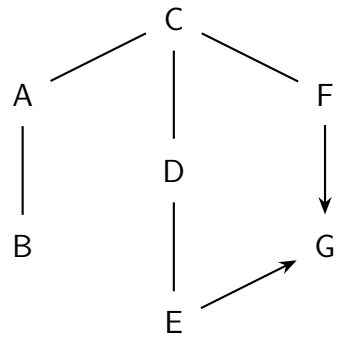


Figure 2: CPDAG of DAG from Figure 1

2 Greedy Equivalence Search

This section will introduce the score-based causal discovery algorithm Greedy Equivalence Search (GES). We will show that the algorithm in the limit of large sample size is sound and complete of the true Markov equivalence class.
We will start by introducing score-based causal discovery, then we will present the Bayesian Information Criterion and its properties. In the last subsection we will define GES and its properties.

A score-based algorithm predicts the graph by comparing different graph structures' ability to fit the data. The idea is that graphs containing wrongful conditional independencies will give bad model fits on the data.

Provided a scoring criterion $S(\mathcal{G}, \mathbf{D})$, the simplest of score-based algorithms is the **Best Scoring Graph** (Peters et al., 2017). Here, we score every graph $\mathcal{G}$ over the nodes $\mathbf{X}$ and choose the graph that yields the maximum score as our prediction.

$$\hat{\mathcal{G}} = \underset{\mathcal{G} \text{ over } \mathbf{X}}{\operatorname{argmax}} S(\mathbf{D}, \mathcal{G}) \tag{1}$$

Which scoring criterion to choose is not given by the method. In the next subsection, we will introduce a scoring criterion and present its properties.

2.1 Bayesian Information Criterion

Here, we will introduce the Bayesian Information Criterion (BIC) (Schwarz, 1978). We will denote it S_B and define it as

$$S_B(\mathcal{G}, \mathbf{D}) = \log p(\mathbf{D}|\hat{\theta}, \mathcal{G}) - \frac{\#\text{parameters}}{2} \log m \tag{2}$$

Here $\hat{\theta}$ denotes the maximum-likelihood values and m is the number of observations in the data $\mathbf{D}$. In some literature, the BIC is negative Equation (2). We have chosen to use this version of BIC since it is similar to the one used by D. M. Chickering (2003) and Kalisch et al. (2024).
We now introduce some definitions of properties that can be held by a scoring criterion.

Definition 1. *(Decomposable score)*
The score S of any DAG $\mathcal{G}$ is decomposable if we can rewrite the score S as a sum of local scores of the nodes given the parents of the nodes

$$S(\mathcal{G}, \mathbf{D}) = \sum_{i=1}^{d} s(X_i, Pa_i^{\mathcal{G}}), \tag{3}$$

where s denotes the local score function, X_i is a node in $\mathcal{G}$ and $Pa_i^{\mathcal{G}}$ is the parents of node X_i in $\mathcal{G}$.

The score function S is computed by all data $\mathbf{D}$ while the local score s will be computed only using data for node X_i and its parents. The data of X_i and $Pa_i^{\mathcal{G}}$ is implied in s. When using a decomposable scoring criterion to compare two DAGs, we only need to compare the score of the terms in Equation (3) for which the corresponding nodes do not share parent sets in the two DAGs. We have by D. M. Chickering (2003) that BIC is decomposable. Hence we can write BIC as:

$$S_B(\mathbf{D}, \mathcal{G}) = \sum_{i=1}^{d} s_B(X_i, Pa_i^{\mathcal{G}}) = \sum_{i=1}^{d} \log p(\mathbf{D}(X_i, Pa_i^{\mathcal{G}})|\hat{\theta}, \mathcal{G}) - \frac{\#\text{parameters}}{2} \log m, \tag{4}$$

where $\mathbf{D}(X_i, Pa_i^{\mathcal{G}})$ is the data of the nodes X_i and $Pa_i^{\mathcal{G}}$. $p(\mathbf{D}(X_i, Pa_i^{\mathcal{G}})|\hat{\theta}, \mathcal{G})$ is the likelihood of the local model defined on X_i and $Pa_i^{\mathcal{G}}$ and #parameters is the number of parameters in this model.

We say that a distribution p is *contained in* a DAG $\mathcal{G}$ if all d-separations statements imply a conditional independence statement in p[1].
For an even stronger relationship between the distribution and the graph, we use the notation *perfect map*. Given a DAG $\mathcal{G}$ and a probability distribution p, we say that $\mathcal{G}$ is a perfect map of p if (1) every independence constraint in p is implied by the structure $\mathcal{G}$[2] and (2) p is contained in a DAG $\mathcal{G}$. Both concepts can be extended from a DAG to an equivalence class $\mathcal{E}$, since an equivalence class $\mathcal{E}$ and a DAG $\mathcal{G} \in \mathcal{E}$ share all the same independence constraints. We may rephrase the assumption of p having dependency structure according to some underlying data-generating DAG $\mathcal{G}$, to the assumption that $\mathcal{G}$ is a perfect map of p.

[1] Often referred to as Markov property
[2] Often referred to as faithfulness

Definition 2. *(Consistent Scoring Criterion)*
Let **D** *be a set of data consisting of m records that are i.i.d. samples from some distribution p. A scoring criterion S is consistent if in the limit as m grows large, the following two properties hold:*

1. *If $\mathcal{G}'$ contains p and $\mathcal{G}$ does not contain p, then $S(\mathcal{G}', \mathbf{D}) > S(\mathcal{G}, \mathbf{D})$*
2. *If $\mathcal{G}'$ and $\mathcal{G}$ both contain p, and $\mathcal{G}$ has fewer parameters than $\mathcal{G}'$, then $S(\mathcal{G}, \mathbf{D}) > S(\mathcal{G}', \mathbf{D})$*

It has been shown by Haughton (1988) that BIC is a consistent scoring criterion for Gaussian and multinomial data.
We now define the concept of being locally consistent.

Definition 3. *(Locally Consistent Scoring Criterion)*
Let **D** *be a set of data consisting of m records that are i.i.d. samples from some distribution p. Let $\mathcal{G}$ be any DAG with no edge $X_i \rightarrow X_j$, and let $\mathcal{G}'$ be the DAG that results from adding the edge $X_i \rightarrow X_j$ to $\mathcal{G}$. A scoring criterion $S(\mathcal{G}, \mathbf{D})$ is locally consistent if the following two properties hold:*

1. *If $X_j \not\perp\!\!\!\perp_p X_i | Pa_j^{\mathcal{G}}$, then $S(\mathcal{G}', \mathbf{D}) > S(\mathcal{G}, \mathbf{D})$*
2. *If $X_j \perp\!\!\!\perp_p X_i | Pa_j^{\mathcal{G}}$, then $S(\mathcal{G}', \mathbf{D}) < S(\mathcal{G}, \mathbf{D})$*

Here, $\perp\!\!\!\perp_p$ denotes being conditional independent in the distribution.

Utilizing that BIC is globally consistent and that BIC is decomposable, it has been proven in Lemma 7 in D. M. Chickering (2003) that BIC is also locally consistent.

So far, we have considered scoring functions S evaluated on DAGs and data, but we will extend the scoring criterion to be an evaluation of an entire equivalence class if the score criterion is *score equivalent*.

Definition 4. *(Score equivalent in the Markov equivalence class)*
We will say a scoring criterion S is score equivalent in the Markov equivalence class $\mathcal{E}$ or just score equivalent if for any pair of DAGs $\mathcal{G}, \mathcal{G}'$ in $\mathcal{E}$

$$S(\mathcal{G}, \mathbf{D}) = S(\mathcal{G}', \mathbf{D})$$

For a score equivalent score criterion the notation $S(\mathcal{E}, \mathbf{D})$ makes sense, even though we in practice will score a $\mathcal{G} \in \mathcal{E}$ to score the equivalence class $\mathcal{E}$.
We have by D. M. Chickering (2003) that BIC is score equivalent in the Markov equivalence

classes.
We have that for a consistent scoring criterion and for $\mathcal{E}$ being a perfect map of the true distribution p that in the limit of large sample size (D. M. Chickering, 2003):

$$S(\mathcal{E}, \mathbf{D}) > S(\mathcal{E}', \mathbf{D}) \text{ for all } \mathcal{E} \neq \mathcal{E}' \tag{5}$$

We assemble all the properties shown in D. M. Chickering (2003) of BIC into a proposition.

Proposition 1. *BIC is decomposable, consistent, locally consistent, and score equivalent on i.i.d. data from a multivariate Gaussian distribution or multinomial distributions for discrete variables*

The case of discrete variables will not be addressed in this thesis.

2.2 Properties of GES

Earlier, we described the algorithm called Best Scoring Graph, where we, by scoring every possible DAG and choosing Equation (1) as our estimator, would estimate the underlying data-generating graph. This would ensure that we find the global maximum, but as we see in Table 1, the number of DAGs that have to be scored grows in a way that makes it computationally infeasible for larger number of nodes.

#Nodes	#DAGs
1	1
2	3
3	25
4	543
5	29281
6	3781503
7	1138779265
⋮	⋮
14	14394281410443983349417907198395351 03

Table 1: The number of DAGs depending on the number of nodes (OEIS Foundation Inc., 2023)

Instead, we will do two things to optimize the number of graphs we need to score. By using a score equivalent score criterion we can make do with only scoring one graph from each of the equivalence classes that we wish to score. This reduces the number of graphs we need to score.

Secondly, we will not score every equivalence class to look for the maximum, but instead go through the equivalence classes step by step until we are unable to improve the score anymore.
We define the Greedy Equivalence Search (GES) which does both of these things and furthermore is sound and complete in the limit of large sample size (D. M. Chickering, 2003).

The GES uses a score equivalent, locally consistent, and decomposable score criterion. This could, for instance, be the BIC score (Proposition 1). In this thesis, we will assume GES to use the BIC as a scoring criterion.
The GES algorithm seeks to maximize the score criterion by doing a greedy search over Markov equivalence classes. The word greedy refers to wanting to maximize the score as much as possible in each step. GES originally consisted of two phases when first proposed by D. M. Chickering (2003). These two phases are the *forward* phase and the *backward* phase. It has since been discovered to improve the prediction when having a third phase called *turning* (Hauser & Bühlmann, 2012).
The forward phase adds edges till a local optimum is reached, the backward phase removes edges till a local optimum is reached, and the turning phase reverses directed edges till a local optimum is reached.

A phase in GES consists of steps which improve the score. Each step operates by evaluating all *neighboring* equivalence classes of the current equivalence class. The set of neighboring equivalence classes is defined for each phase as follows:
$\mathcal{E}'$ is in the set of forward step neighbouring equivalence classes of $\mathcal{E}$, denoted $\boldsymbol{\mathcal{E}}^{+}(\mathcal{E})$, if and only if there exists a $\mathcal{G} \in \mathcal{E}$ and a $\mathcal{G}' \in \mathcal{E}'$ such that $\mathcal{G}$ becomes $\mathcal{G}'$ by one single directed edge addition.
$\mathcal{E}'$ is in the set of backward step neighbouring equivalence classes of $\mathcal{E}$, denoted $\boldsymbol{\mathcal{E}}^{-}(\mathcal{E})$, if and only if there exists a $\mathcal{G} \in \mathcal{E}$ and a $\mathcal{G}' \in \mathcal{E}'$ such that $\mathcal{G}$ becomes $\mathcal{G}'$ by one single directed edge removal.
$\mathcal{E}'$ is in the set of turning step neighbouring equivalence classes of $\mathcal{E}$, denoted $\boldsymbol{\mathcal{E}}^{\leftrightarrow}(\mathcal{E})$, if and only if there exist a $\mathcal{G} \in \mathcal{E}$ and a $\mathcal{G}' \in \mathcal{E}'$ such that $\mathcal{G}$ becomes $\mathcal{G}'$ by one single directed edge reversal.

GES initiates with the equivalence class of the empty DAG $\mathcal{E}$, hereby meaning that there are no edges. Then, the forward phase commences and repeatedly replaces $\mathcal{E}$ with the equivalence class from $\boldsymbol{\mathcal{E}}^{+}(\mathcal{E})$ with the highest improvement of score. The forward phase keeps replacing until no equivalence class from $\boldsymbol{\mathcal{E}}^{+}(\mathcal{E})$ has a higher score than $\mathcal{E}$. Then the backward phase repeatedly replaces $\mathcal{E}$ with the equivalence class from $\boldsymbol{\mathcal{E}}^{-}(\mathcal{E})$ with the highest improvement of score, and do so until no equivalence class from $\boldsymbol{\mathcal{E}}^{-}(\mathcal{E})$ has a higher score than $\mathcal{E}$. Then the turning phase repeatedly replaces $\mathcal{E}$ with the equivalence class from $\boldsymbol{\mathcal{E}}^{\leftrightarrow}(\mathcal{E})$ with the highest improvement of score, and do so until no equivalence class from $\boldsymbol{\mathcal{E}}^{\leftrightarrow}(\mathcal{E})$ has a higher score than $\mathcal{E}$. If there were made any change

by either the forward, backward, or turning phase, $\mathcal{E}$ is passed on to the forward phase again, and we run through all 3 phases again.
For an overlook of the pseudo code of GES we refer to Algorithm 1 below.

Algorithm 1 Greedy Equivalence Search (GES)

Initialize:
$\mathcal{E} \leftarrow$ the equivalence class of the empty DAG (no edges).
Compute initial score $S_B(\mathcal{E}, \mathbf{D})$.
while there are changes that improve the score **do**
    Forward phase
    for all $\mathcal{E}' \in \boldsymbol{\mathcal{E}}^+(\mathcal{E})$ **do**
        Compute $S_B(\mathcal{E}', \mathbf{D})$.
    end for
    if any score of $\mathcal{E}'$ is bigger **then**
        $\mathcal{E} \leftarrow \underset{\mathcal{E}' \in \boldsymbol{\mathcal{E}}^+(\mathcal{E})}{\operatorname{argmax}} (S_B(\mathcal{E}', \mathbf{D}))$
    end if
    Backward phase
    for all $\mathcal{E}' \in \boldsymbol{\mathcal{E}}^-(\mathcal{E})$ **do**
        Compute $S_B(\mathcal{E}', \mathbf{D})$.
    end for
    if any score of $\mathcal{E}'$ is bigger **then**
        $\mathcal{E} \leftarrow \underset{\mathcal{E}' \in \boldsymbol{\mathcal{E}}^-(\mathcal{E})}{\operatorname{argmax}} (S_B(\mathcal{E}', \mathbf{D}))$
    end if
    Turning phase
    for all $\mathcal{E}' \in \boldsymbol{\mathcal{E}}^{\leftrightarrow}(\mathcal{E})$ **do**
        Compute $S_B(\mathcal{E}', \mathbf{D})$.
    end for
    if any score of $\mathcal{E}'$ is bigger **then**
        $\mathcal{E} \leftarrow \underset{\mathcal{E}' \in \boldsymbol{\mathcal{E}}^{\leftrightarrow}(\mathcal{E})}{\operatorname{argmax}} (S_B(\mathcal{E}', \mathbf{D}))$
    end if
end while
Output: $\mathcal{E}$

Here, we will introduce two lemmas by D. M. Chickering (2003), which together are analogous to stating that GES is sound and complete in the limit of large sample size. We will also sometimes refer to being in the limit of large sample size as being in the oracle setting or oracle scenario.

Before we introduce the two lemmas, we state a theorem from D. M. Chickering (2003) that we will not prove but instead refer to D. M. Chickering (2003) for the proof.

Theorem 1. *(D. M. Chickering, 2003) Let $\mathcal{G}$ and $\mathcal{H}$ be any pair of DAGs such that $\mathcal{G} \leq \mathcal{H}$. Let r be the number of edges in $\mathcal{H}$ that have opposite orientation in $\mathcal{G}$, and let m be the number of edges in $\mathcal{H}$ that do not exist in either orientation in $\mathcal{G}$. There exists a sequence of at most r + 2m edge reversals and additions in $\mathcal{G}$ with the following properties:*

1. *Each edge reversed is a covered edge*
2. *After each reversal and addition $\mathcal{G}$ is a DAG and $\mathcal{G} \leq \mathcal{H}$*
3. *After all reversals and additions $\mathcal{G} = \mathcal{H}$*

where $\mathcal{G} \leq \mathcal{H}$ refer to that every independence relationship in $\mathcal{H}$ holds in $\mathcal{G}$ and a covered edge is an edge that in its Markov equivalence class cannot be directed.

We also note that as there exist a data-generating DAG $\mathcal{G}$ which is a perfect map of p, we have, as noted in D. M. Chickering (2003), that any such distribution p must obey the composition independence axiom described in Pearl (1988): if $X \not\perp\!\!\!\perp_p \mathbf{Y}|\mathbf{Z}$, then there exists a singleton element $Y \in \mathbf{Y}$ such that $X \not\perp\!\!\!\perp_p Y|\mathbf{Z}$.
We will state the lemmas and prove them to be true by the use of this observation and Theorem 1.

Lemma 1. *(Optimality of forward phase, D. M. Chickering, 2003)*
Let $\mathcal{E}$ be the equivalence class that is the result from doing the forward phase in GES, let p be the distribution from which the data **D** *was generated and m be the number of observations in* **D**. *Then in the limit of large m, p is contained in $\mathcal{E}$*

Proof. We prove this lemma by contradiction:
We assume for contradiction that p is not contained in $\mathcal{E}$. Let $\mathcal{G} \in \mathcal{E}$. Because p is not contained in $\mathcal{E}$ and $\mathcal{G}$ contains the same independence structure as all other graphs in $\mathcal{E}$ we know there exist at least one node X_i such that $X_i \not\perp\!\!\!\perp_p \mathbf{Y}|Pa_i^{\mathcal{G}}$ where $\mathbf{Y}$ is the set of non-descendants of X_i. By the Markov property of $\mathcal{G}$ we know that the set $Pa_i^{\mathcal{G}}$ d-seperates X_i from $\mathbf{Y}$. Since the composition independence axiom holds for p we know that there is a singleton $Y \in \mathbf{Y}$ such that $X_i \not\perp\!\!\!\perp_p Y|Pa_i^{\mathcal{G}}$. Because our score function is locally consistent, we have that adding the edge $Y \longrightarrow X_i$, which does not introduce any cycles since Y is a non-descendant of X_i, will increase the score. A graph $\mathcal{G}'$ identical to $\mathcal{G}$ but with the new edge is clearly contained in $\mathcal{E}^+(\mathcal{E})$ and therefore $\mathcal{E}$ can not be a local maximum, and we have a contradiction. □

Lemma 2. *(Optimality of backward phase, D. M. Chickering, 2003)*
Let $\mathcal{E}$ be the equivalence class that is the result from doing the backward phase in GES starting out with a $\mathcal{E}$ containing p, where p is the distribution from which the data **D** *was generated and m the number of observations in* **D**. *Then, in the limit of large m, $\mathcal{E}$ is a perfect map of p.*

Proof. First, we realize that each backward step ensures that the new equivalence class also contains p. If the first backward step of the phase is such that p is no longer contained, then the score must decrease since the scoring function is consistent. Since the backward step is greedy and seeks to maximize, this can not happen.
The rest of the proof will be proof by contradiction:
We assume the backward phase to result in $\mathcal{E}$ where $\mathcal{E}$ is sub-optimal and contains p. Let $\mathcal{E}'$ be the perfect map of p. Then we know by Equation (5) that $S(\mathcal{E}', \mathbf{D}) > S(\mathcal{E}, \mathbf{D})$. Now let $\mathcal{G}$ be any DAG in $\mathcal{E}$ and $\mathcal{G}'$ be any dag in $\mathcal{E}'$. We have that any independence relationship that holds in $\mathcal{G}$ holds in $\mathcal{G}'$. We also know $\mathcal{G}$ contains more edges than $\mathcal{G}'$, since every independence relationship in $\mathcal{G}$ holds in $\mathcal{G}'$ and by assumption $\mathcal{E} \neq \mathcal{E}'$. By Theorem 1, we can transform $\mathcal{G}'$ into $\mathcal{G}$ by reversing and adding edges while ensuring that any independence relationship is contained in all the steps. Since $\mathcal{G}$ has more edges than $\mathcal{G}'$, we know there to be at least one edge addition. Consider the DAG $\mathcal{G}^*$ that precedes the very last edge addition in the sequence. Then we have that $\mathcal{E}(\mathcal{G}^*) \in \mathcal{E}^-(\mathcal{E})$. Then, $\mathcal{E}$ can not be a local maximum since $\mathcal{G}^*$ has fewer edges than $\mathcal{G}$, and we know that the score function is consistent. □

As seen by Lemmas 1 and 2 with the assumption of enough data (oracle scenario), it is sufficient to run through forward and backward phase just once to estimate the correct equivalence class. In practice, we will rarely be so lucky, and therefore, adding the turning phase and iterating over the phases until no further improvement of the score may improve our estimate by decreasing statistical errors (Hauser & Bühlmann, 2012).

3 GES in a temporal setting

We will start off by giving a quick introduction to Meek's orientation rules, which will be used in this section. We will then introduce tiered background knowledge in Section 3.1 and then present two causal discovery methods: Simple TGES in Section 3.2 and TGES in Section 3.4. To define TGES, we will first introduce a new scoring criterion, TBIC, in Section 3.3.

We will say that a PDAG contains all DAGs, which have the same skeleton and directed edges as the PDAG. Meek's orientation rules are four rules[3] that infer *maximality* in a PDAG, by directing undirected edges in the PDAG. Maximality ensures that no more undirected edges can be directed in the PDAG while the PDAG contains the same set of DAGs. When a PDAG is maximal, we will say that it is a *Maximally oriented Partially Directed Acyclic Graph* (MPDAG).

3.1 Background knowledge

This subsection introduces and defines general *background knowledge* for graphs, and more specifically *tiered background knowledge* for graphs.

We define background knowledge $\mathcal{K} = (\mathcal{R}, \mathcal{F})$ as a set of required directed edges $\mathcal{R}$ and forbidden directed edges $\mathcal{F}$. In this framework, an undirected edge is defined as two directed edges on the same nodes in each direction.
We will say that a DAG $\mathcal{G}$ *encodes* the background knowledge $\mathcal{K}$ if all directed edges in $\mathcal{R}$ and none of the directed edges in $\mathcal{F}$ are present in $\mathcal{G}$. If a graph $\mathcal{G}$ does not encode $\mathcal{K}$, we will say that $\mathcal{G}$ contradicts $\mathcal{K}$. A similar concept is introduced for a CPDAG. We will say a CPDAG/equivalence class $\mathcal{E}$ is *in agreement* with background knowledge $\mathcal{K}$ if there exists a $\mathcal{G} \in \mathcal{E}$ that encodes $\mathcal{K}$, and not in agreement if all $\mathcal{G} \in \mathcal{E}$ contradicts $\mathcal{K}$.

As described earlier, a Completed Partially Directed Acyclic Graph (CPDAG) represents the DAGs (Directed Acyclic Graph) that belong to the Markov equivalence class. A Markov equivalence class $\mathcal{E}$ restricted by background knowledge $\mathcal{K}$, denoted $\mathcal{E}^{\mathcal{K}}$, will only contain DAGs that not just all have the same conditional independencies but also all encode $\mathcal{K}$. We will denote a Markov equivalence class $\mathcal{E}$ restricted by background knowledge $\mathcal{K}$ as $\mathcal{E}^{\mathcal{K}}$.

In this thesis, we will focus on a specific type of background knowledge, namely *tiered background knowledge* $\mathcal{K} = (\mathcal{R}, \mathcal{F})$ where $\mathcal{R} = \emptyset$ and $\mathcal{F}$ depends on the *tiered ordering* defined below.

[3] Graphical representation of the four rules are located in the Appendix A.2.

Definition 5. *(Tiered ordering). Let $\mathcal{G}$ be a PDAG with node set $\mathbf{X}$ of size d, and let $T \in \mathbb{N}, T \leq d$. A tiered ordering of the nodes in $\mathbf{X}$ is a map $\tau : \mathbf{X} \mapsto \{1, ..., T\}^d$ that assign each node $X \in \mathbf{X}$ to a unique tier $t \in \{1, ..., T\}$.*

We will define tiered background knowledge from the definition of a tiered ordering

Definition 6. *(Tiered background knowledge)*
Given a tiered ordering τ we define tiered background knowledge as $\mathcal{K} = (\mathcal{R}, \mathcal{F})$ where $\mathcal{R} = \emptyset$ and $\mathcal{F}$ is the set of directed edges $\{A \to B\}$ such that $\tau(A) > \tau(B)$.

From here on, all background knowledge $\mathcal{K}$ in this thesis will be assumed to be tiered, and we will refer to a restricted equivalence class when referring to it being restricted by tiered background knowledge.

We have that graphs restricted by tiered background knowledge are made up of two types of edges:

Definition 7. *(cross-tier and in-tier edge) Let $\mathcal{H} = (V, \mathbf{E})$ be a PDAG with the node set $\mathbf{X}$, the set of directed edges $\mathbf{E}$ and τ a tiered ordering of $\mathbf{X}$.*
*A directed edge $\{A \to B\} \in \mathbf{E}$ is a **cross-tier edge** (relative to τ) if $\tau(A) < \tau(B)$.*
*A directed edge $\{A \to B\} \in \mathbf{E}$ is a **in-tier edge** (relative to τ) if $\tau(A) = \tau(B)$.*

A tiered ordering does not imply the existence of an edge but only the absence of a direction of an edge. Hence, we do not restrict anything about the skeleton of the graph, only about the direction given there exists an edge in the graph.

Like we have used the CPDAG to represent an equivalence class, we will now introduce a new type of graph which represents a restricted equivalence class $\mathcal{E}^{\mathcal{K}}$, a tiered Maximally oriented Partially Directed Acyclic Graph (tiered MPDAG) (Bang & Didelez, 2023). A tiered MPDAG has a directed edge wherever all DAGs in $\mathcal{E}^{\mathcal{K}}$ agree on the orientation of an edge, and the tiered MPDAG has an undirected edge wherever two DAGs in $\mathcal{E}^{\mathcal{K}}$ disagree on the orientation of the edge.
We may restrict an equivalence class $\mathcal{E}$, that is in agreement with the tiered background knowledge $\mathcal{K}$, according to $\mathcal{K}$ by orienting all the cross-tier edges according to $\mathcal{K}$. See Figure 3 for an example of the CPDAG from Figure 2 having been restricted according to $\mathcal{K}$ with $\tau(A) = \tau(B) = 1, \tau(C) = \tau(D) = \tau(E) = 2, \tau(F) = \tau(G) = 3$, where all newly oriented edges are blue. The newly oriented cross-tier edges will give cause to orient other edges. We can do so by orienting according to Meek's rule 1 (Meek, 1995) (See Appendix A.2). Meek's rule 1 prohibits new v-structures in the graph. The resulting graph will be a tiered MPDAG (Bang & Didelez, 2023). See the example in Figure 4 where edges oriented according to Meek's rule 1 are blue.

It was shown in Bang and Didelez (2023) that using Meek's rule 1 is sufficient for maximality when restricting according to tiered background knowledge, while for general background knowledge, Meek's rules 1-4 are all needed (See Appendix A.2). Meek's rules 2-4 prohibit cycles, which we are guaranteed not to introduce by restricting with tiered background knowledge (Bang & Didelez, 2023).

Throughout this thesis, we assume that our tiered background knowledge is encoded by the true distribution p. Hence, we are never using incorrect background knowledge.

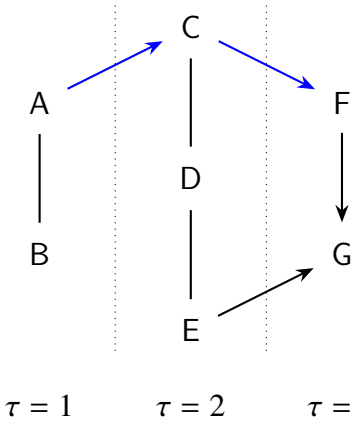


Figure 3: Tiered PDAG of CPDAG from Figure 2. Edges oriented by tiered background knowledge are blue.

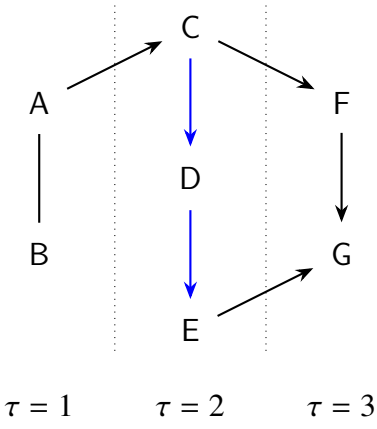


Figure 4: Tiered MPDAG of CPDAG from Figure 2. Edges oriented according to Meek's rule 1 are blue.

3.2 Simple Temporal Greedy Equivalence Search

We wish to be able to use tiered background knowledge when we are performing causal discovery, because we suspect that tiered background knowledge leads to considerable gains in informativeness and computational efficiency. As mentioned earlier we wish to develop a tiered extension of GES.

The question of how to utilize the tiered background knowledge when predicting a graph with GES is not obvious, and several approaches seem viable.
The issue of incorporating general background knowledge into a GES algorithm has been addressed before by other authors, but not tiered background knowledge like described in the previous sub-

section. The authors behind the pcalg package (Kalisch et al., 2024) in R have made it possible to constrict the search space to avoid specified adjacencies of a CPDAG. Even though their aim seems to have been to implement a hybrid algorithm of PC and GES, the implementation allows for feeding background knowledge into the GES algorithm.
The authors behind the Knowledge-guided Greedy Equivalence Search algorithm (KGS by Hasan & Gani, 2023) have made a tool implemented on top of the causal-learn package in Python, which is an extension of the Tetrad Java code (Zheng et al., 2024). Like pcalg, KGS allows for incorporating the absence of an adjacency, but also the existence of both a directed edge and an undirected edge.

In both cases the background knowledge implemented does not allow for handling the information induced from tiered background knowledge. In KGS and pcalg we can only completely rule out the existence of an adjacency or say with certainty that an edge exists. Tiered background knowledge wish to direct an edge given that the adjacency exists, while not being certain of the existence of the adjacency. This motivates the need for a new version of the GES algorithm that takes tiered background knowledge into account.

In this section, we propose the new algorithm *Simple Temporal Greedy Equivalence Search* (Simple TGES).
We will go through how the new algorithm was implemented, show it is optimal and in a later section (Section 5) we will test the performance of it. The code for running Simple TGES can be found in the script ”Simulate_functions.R” and function ”ges_to_simple_tges_adj()” (Larsen, 2024).

The idea behind Simple TGES is presented in Figure 5, and the pseudo code is presented in Algorithm 2.
Simple TGES initiates by running the GES algorithm to completion. GES results in a CPDAG, which we are not guaranteed is in agreement with $\mathcal{K}$ when outside the oracle scenario. We, therefore, remove all directed edges that contradict $\mathcal{K}$, which leaves us with a PDAG that is in agreement with $\mathcal{K}$. Then, this PDAG is restricted according to tiered background knowledge $\mathcal{K}$ and results in another PDAG. We then obtain a tiered MPDAG[(Remark 1)] by using Meek’s rules 1-4 (See Appendix A.2). In the oracle scenario, the stage that removes all directed edges that contradict $\mathcal{K}$ and Meek’s rules 2-4 are redundant.

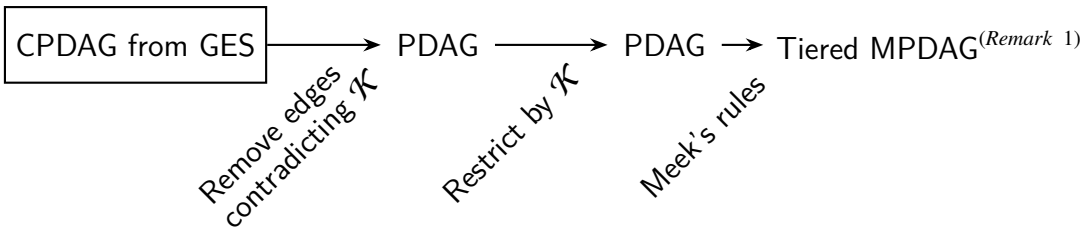


Figure 5: Simple TGES: This is an overview of how the method Simple TGES works. The square represents the entire process of GES, and the CPDAG, of which it terminates with, is passed on. $\mathcal{K}$ represents the tiered background knowledge, and $\mathcal{E}$ represents Markov equivalence class

Remark 1. *A tiered MPDAG is defined from a CPDAG that is in agreement with the background knowledge $\mathcal{K}$ (Bang & Didelez, 2023), since there is no guarantee outside the oracle setting that $\mathcal{E}$ from GES is in agreement with $\mathcal{K}$, we cannot be sure this is an actual tiered MPDAG.*

We shall see later in this thesis that Remark 1 will have an influence on the performance of Simple TGES outside of the oracle setting. Meanwhile, we will see that the Simple TGES is optimal in the limit of large sample size.

We will not use the same definition of optimality as in Lemmas 1 and 2, but instead extend the definition to tiered optimality which is defined as sound, complete and all DAGs in the class encode $\mathcal{K}$.

Theorem 2. *(Tiered optimality of Simple TGES)*
Let $\mathcal{E}^{\mathcal{K}}$ be the restricted equivalence class that is a result from running Simple TGES, let p be the distribution from which the data **D** *was generated, m be the number of observations in* **D** *and $\mathcal{K}$ the tiered background knowledge of p. Then in the limit of large m, $\mathcal{E}^{\mathcal{K}}$ is a perfect map of p and all DAGs in $\mathcal{E}^{\mathcal{K}}$ encode $\mathcal{K}$.*

Proof. Assume that $\mathcal{E}$ is the equivalence class that is a result of running GES to completion. By Lemmas 1 and 2 we have that $\mathcal{E}$ is a perfect map of p. Since $\mathcal{K}$ is assumed to have encoded the data-generating DAG $\mathcal{G}$ from which p stems from, and we have that $\mathcal{G} \in \mathcal{E}$, we know that $\mathcal{E}$ is in agreement with $\mathcal{K}$. Hence, we are not removing any edges in the CPDAG corresponding to $\mathcal{E}$. $\mathcal{E}^{\mathcal{K}}$ have the same v-structures and skeleton as $\mathcal{E}$ which means that $\mathcal{E}^{\mathcal{K}}$ is a perfect map of p and by design all DAGs in $\mathcal{E}^{\mathcal{K}}$ encode $\mathcal{K}$. □

3.2.1 Implementation

We have implemented Simple TGES in R as the function ”ges_to_simple_tges_adj()” (Larsen, 2024). The function takes in a list that contains an ”essgraph” class. This type of list is returned by the ”ges()” function from pcalg (Kalisch et al., 2024). The Simple TGES function then removes all directional edges contradicting $\mathcal{K}$ and then uses Meek's rules to infer edge orientations. The result is then returned as an adjacency matrix.

The pseudo code for the algorithm is provided below.

Algorithm 2 Simple TGES

1: Run GES and obtain equivalence class $\mathcal{E}$ and corresponding CPDAG
2: Remove edges contradicting the tiered background knowledge and obtain a PDAG $\mathcal{G}$
3: Direct undirected edges in $\mathcal{G}$ according to $\mathcal{K}$
4: **while** no further edges can be directed of the PDAG $\mathcal{G}$ **do**
5: Use Meek's rule 1-4 on $\mathcal{G}$
6: **end while**
7: **Output:**
8: The tiered MPDAG $\mathcal{G}$

3.3 Temporal Bayesian Information Criterion (TBIC)

In this section, we will introduce a new scoring criterion, *Temporal Bayesian Information Criterion* (TBIC). We show that the new scoring criterion is score equivalent. Then we propose a new property, being consistent according to tiered background knowledge, and we prove that this property is held by the scoring criterion. We will also show that TBIC is decomposable, score equivalent and optimal when scoring the restrictive equivalence class that is a perfect map of our generative distribution p.

Whereas we saw in Theorem 2 that Simple TGES is tiered optimal in the oracle scenario, it has been noted by Petersen et al. (2021) that a post-hoc method, like Simple TGES, is a less attractive choice when we have high trust in the validity of our tiered background knowledge. In a post-hoc method, we learn the graph from the data first and then make it comply with background knowledge. Instead, we wish to incorporate the tiered background knowledge into the algorithm and learn as much as we can about the graph from the data given the tiered background knowledge.
We also wish to ensure that our algorithm allows for avoiding the issue addressed in Remark 1.

To do this, we first present the new scoring criterion TBIC. The scoring criterion is defined and implemented in R, and the code is provided in the script "TGES_function.R" (Larsen, 2024).

While a score equivalent score criterion is essential for implementing an efficient algorithm that will estimate the equivalence class $\mathcal{E}$ of the underlying data-generating graph, it does not help us when the goal is to estimate something more specific such as a restricted equivalence class $\mathcal{E}^{\mathcal{K}}$. We wish to have a scoring criterion that can differentiate between a DAG $\mathcal{G} \in \mathcal{E}^{\mathcal{K}}$ and a DAG $\mathcal{G}' \in \mathcal{E}$ where $\mathcal{G}' \notin \mathcal{E}^{\mathcal{K}}$.

We introduce the new scoring criterion called Temporal Bayesian Information Criterion (TBIC), which takes tiered background knowledge into account. The word temporal is used since we, in practice, most often would have tiers made from a temporal setting. We will denote TBIC as $\tilde{S}$. The score criterion is equal to the Bayesian Information Criterion S_B if the graph encodes the background knowledge and otherwise it is set to $-\infty$.

Definition 8. *(Temporal Bayesian Information Criterion)*
For tiered background knowledge $\mathcal{K}$

$$\tilde{S}(\mathcal{G}, \mathbf{D}, \mathcal{K}) = \begin{cases} S_B(\mathcal{G}, \mathbf{D}) & \text{if } \mathcal{G} \text{ encodes } \mathcal{K} \\ -\infty & \text{if } \mathcal{G} \text{ contradicts } \mathcal{K} \end{cases}$$

The intuitive sense of TBIC is to create a score criterion that penalizes graphs that do not encode $\mathcal{K}$ in a way that makes it impossible for a score-based algorithm to terminate with a graph that is not in agreement with $\mathcal{K}$.
We note that the definition of TBIC could be extended to any type of background knowledge.

3.3.1 Properties of TBIC

In this subsection we will introduce a new definition of consistency for scoring criteria with background knowledge. We will show that TBIC has this property. Afterwards we will prove that TBIC also has the properties score equivalence, decomposability and it has its optimal solution in the restricted equivalence class that is a perfect map of p. These properties will prove useful for developing an extended version of Simple TGES.

Definition 9. *(Consistent scoring criterion according to tiered background knowledge $\mathcal{K}$)*
Let **D** *be a set of data consisting of m records that are i.i.d. samples from some distribution p that encodes the tiered background knowledge $\mathcal{K}$. A scoring criterion S is consistent according to tiered background knowledge $\mathcal{K}$ if in the limit as m grows large, the following holds:*

1. *If $\mathcal{G}'$ and $\mathcal{G}$ both encodes $\mathcal{K}$ and we have that $\mathcal{G}'$ contains p and $\mathcal{G}$ does not contain p, then* $S(\mathcal{G}', \mathbf{D}, \mathcal{K}) > S(\mathcal{G}, \mathbf{D}, \mathcal{K})$
2. *If $\mathcal{G}'$ and $\mathcal{G}$ both encodes $\mathcal{K}$ and both contain p, and $\mathcal{G}$ has fewer parameters than $\mathcal{G}'$, then* $S(\mathcal{G}, \mathbf{D}, \mathcal{K}) > S(\mathcal{G}', \mathbf{D}, \mathcal{K})$
3. *If $\mathcal{G}'$ encodes $\mathcal{K}$ and $\mathcal{G}$ contradicts $\mathcal{K}$ then* $S(\mathcal{G}', \mathbf{D}, \mathcal{K}) > S(\mathcal{G}, \mathbf{D}, \mathcal{K})$

The case of two graphs not encoding $\mathcal{K}$ is not interesting as both are wrong.
We will show that TBIC holds this property.

Proposition 2. *TBIC is a consistent scoring criterion according to tiered background knowledge $\mathcal{K}$*

Proof. 1 & 2: If $\mathcal{G}$ and $\mathcal{G}'$ encode the tiered background knowledge $\mathcal{K}$ then the scoring criterion TBIC is equal to the scoring criterion *BIC* which by Proposition 1 has property 1 and 2.
3: Assume $\mathcal{G}'$ encode $\mathcal{K}$ and $\mathcal{G}$ contradicts $\mathcal{K}$ then the property follows from the fact that $\tilde{S}(\mathcal{G}', \mathbf{D}, \mathcal{K}) = S_B(\mathcal{G}', \mathbf{D}) > -\infty$ and $\tilde{S}(\mathcal{G}, \mathbf{D}, \mathcal{K}) = -\infty$. □

Next, let us introduce another important property of TBIC, namely that it is score equivalent over any $\mathcal{G} \in \mathcal{E}^{\mathcal{K}}$. This will prove to be important when wishing to use TBIC as a scoring criterion in practice. We note that by definition the restricted equivalence class is a subset of the equivalence class that is not restricted by tiered background knowledge $\mathcal{K}$, $\mathcal{E}^{\mathcal{K}} \subseteq \mathcal{E}$.

Proposition 3. *(TBIC is score equivalent in the restricted equivalence class)*
For any DAGs $\mathcal{G}, \mathcal{G}' \in \mathcal{E}^{\mathcal{K}}$, we have that the TBIC score $\tilde{S}(\mathcal{G}, \mathbf{D}, \mathcal{K}) = \tilde{S}(\mathcal{G}', \mathbf{D}, \mathcal{K})$

Proof. Both $\mathcal{G}$ and $\mathcal{G}'$ encodes $\mathcal{K}$ hence $\tilde{S}(\mathcal{G}, \mathbf{D}, \mathcal{K}) = S_B(\mathcal{G}, \mathbf{D})$ and $\tilde{S}(\mathcal{G}', \mathbf{D}, \mathcal{K}) = S_B(\mathcal{G}', \mathbf{D})$ by definition.
We have that the corresponding equivalence class $\mathcal{E}$ to the restrictive equivalence class $\mathcal{E}^{\mathcal{K}}$ contains $\mathcal{G}, \mathcal{G}' \in \mathcal{E}$ since $\mathcal{E}^{\mathcal{K}} \subseteq \mathcal{E}$. We have by the score equivalence of BIC (Proposition 1) that $S_B(\mathcal{G}, \mathbf{D}) = S_B(\mathcal{G}', \mathbf{D})$ □

This property of TBIC gives sense to using the notation $S(\mathcal{E}^{\mathcal{K}}, \mathbf{D})$ when we are scoring a restricted equivalence class $\mathcal{E}^{\mathcal{K}}$ by scoring a DAG $\mathcal{G} \in \mathcal{E}^{\mathcal{K}}$ with $S(\mathcal{G}, \mathbf{D}, \mathcal{K})$.
The last property of TBIC we wish to state is the property of decomposability. First, we define the local score function of TBIC $\tilde{s}$

$$\tilde{s}(X_i, Pa_i^{\mathcal{G}}, \mathcal{K}) = \begin{cases} s_B(X_i, Pa_i^{\mathcal{G}}) & \text{if } \forall A \in Pa_i^{\mathcal{G}} \text{ we have that } \tau(X_i) \geq \tau(A) \\ -\infty & \text{if } \exists A \in Pa_i^{\mathcal{G}} \text{ such that } \tau(X_i) < \tau(A) \end{cases} \tag{6}$$

Here, s_B is the local score of BIC as seen in Equation (4).

Proposition 4. *(TBIC is decomposable)*
We can write $\tilde{S}$ *as a sum of local scores* $\tilde{s}$ *of the nodes given the parents.* $\tilde{s}$ *is given in Equation* (6).

$$\tilde{S}(\mathcal{G}, \mathbf{D}, \mathcal{K}) = \sum_{i=1}^{d} \tilde{s}(X_i, Pa_i^{\mathcal{G}}, \mathcal{K}) \tag{7}$$

Proof. We separate this proof into two cases: when $\mathcal{G}$ encodes $\mathcal{K}$ and when $\mathcal{G}$ contradicts $\mathcal{K}$.

First let us assume that $\mathcal{G} = (V, \mathbf{E})$ encodes $\mathcal{K}$. Then $\forall$ edges $\{A \rightarrow B\} \in \mathbf{E}$ we have that $\{A \rightarrow B\} \notin \mathcal{F}$, where $\mathcal{F}$ is the set of forbidden edges from $\mathcal{K}$. Hence $\forall \{A \rightarrow B\} \in \mathbf{E}$ we have that $\tau(A) \leq \tau(B)$, which is equivalent to stating that for all $X_i \in \mathbf{X}$ it must be that $\forall A \in Pa_i^{\mathcal{G}}$ have that $\tau(A) \leq \tau(X_i)$. Thus we have that $\tilde{s}(X_i, Pa_i^{\mathcal{G}}, \mathcal{K}) = s_B(X_i, Pa_i^{\mathcal{G}})$.
Then

$$\tilde{S}(\mathcal{G}, \mathbf{D}, \mathcal{K}) = S_B(\mathcal{G}, \mathbf{D}) = \sum_{i=1}^{d} s_B(X_i, Pa_i^{\mathcal{G}}) = \sum_{i=1}^{d} \tilde{s}(X_i, Pa_i^{\mathcal{G}}, \mathcal{K}).$$

Now let us assume that $\mathcal{G}$ contradicts $\mathcal{K}$. Then there $\exists \{A \rightarrow B\} \in \mathbf{E}$ such that $\{A \rightarrow B\} \in \mathcal{F}$. Hence $\exists \{A \rightarrow B\} \in \mathbf{E}$ such that $\tau(A) > \tau(B)$, which is equivalent to stating that there exist a $X_q \in \mathbf{X}$ where it must be that $\exists A \in Pa_q^{\mathcal{G}}$ such that $\tau(A) > \tau(X_q)$. Thus there for the node X_q we have that $\tilde{s}(X_q, Pa_q^{\mathcal{G}}, \mathcal{K}) = -\infty$
Then

$$\sum_{i=1}^{d} \tilde{s}(X_i, Pa_i^{\mathcal{G}}, \mathcal{K}) = \sum_{i=1}^{q-1} \tilde{s} + \sum_{i=q+1}^{d} \tilde{s} + \tilde{s}(X_q, Pa_q^{\mathcal{G}}, \mathcal{K}) = \sum_{i=1}^{q-1} \tilde{s} + \sum_{i=q+1}^{d} \tilde{s} - \infty \overset{*}{=} -\infty = \tilde{S}(\mathcal{G}, \mathbf{D}, \mathcal{K}),$$

where we shortened $\tilde{s}(X_i, Pa_i^{\mathcal{G}}, \mathcal{K})$ as $\tilde{s}$ for easier notation. $*$ is true since all $\tilde{s}(X_i, Pa_i^{\mathcal{G}}, \mathcal{K}) < \infty$. □

We will introduce another property of TBIC, namely that the restricted equivalence class $\mathcal{E}^{\mathcal{K}}$ that is a perfect map of the p is the optimal solution. We know that this restricted equivalence class exists since we have assumed there to exist a true data-generating DAG which is a perfect map of p, and we have assumed that this DAG encodes $\mathcal{K}$.

Proposition 5. *(The $\mathcal{E}^{\mathcal{K}}$ that is a perfect map of the p is the optimal solution)*
Let $\mathcal{E}^{\mathcal{K}*}$ denote the restricted equivalence class that is a perfect map of p (hence encodes $\mathcal{K}$) and let m denote the sample size of $\mathbf{D}$*. Then in the limit of large m*

$$\tilde{S}(\mathcal{E}^{\mathcal{K}*}, \mathbf{D}) > \tilde{S}(\mathcal{E}^{\mathcal{B}}, \mathbf{D}) \text{ for all } \mathcal{E}^{\mathcal{K}*} \neq \mathcal{E}^{\mathcal{B}}$$

Proof. First, we realize that this also holds if $\mathcal{E}^{\mathcal{B}}$ does not encode $\mathcal{K}$. Assume $\mathcal{E}^{\mathcal{B}}$ to not encode $\mathcal{K}$ then for all $\mathcal{G} \in \mathcal{E}^{\mathcal{B}}$ we have $\tilde{S}(\mathcal{G}, \mathbf{D}) = -\infty$ and since $\mathcal{E}^{\mathcal{K}*}$ encodes $\mathcal{K}$ it is strictly greater.
Now assume $\mathcal{E}^{\mathcal{B}}$ encodes $\mathcal{K}$. First notice that if $\mathcal{E}^{\mathcal{K}*} \neq \mathcal{E}^{\mathcal{B}}$ then $\mathcal{E}^* \neq \mathcal{E}$ where $\mathcal{E}^*$ is the Markov equivalence class which by restricting according to $\mathcal{K}$ becomes $\mathcal{E}^{\mathcal{K}*}$, and $\mathcal{E}$ is the Markov equivalence class which by restricting according to $\mathcal{K}$ becomes $\mathcal{E}^{\mathcal{B}}$. We have that $\mathcal{E}^{\mathcal{K}*} \subseteq \mathcal{E}^*$ and $\mathcal{E}^{\mathcal{B}} \subseteq \mathcal{E}$. Also, realize that $\mathcal{E}^*$ is also a perfect map of p since it shares the same conditional independence statements as $\mathcal{E}^{\mathcal{K}*}$. We have by Equation (5) and the definition of $\tilde{S}$ that in the limit of large m

$$\tilde{S}(\mathcal{E}^{\mathcal{K}*}, \mathbf{D}) = S_B(\mathcal{E}^*, \mathbf{D}) > S_B(\mathcal{E}, \mathbf{D}) = \tilde{S}(\mathcal{E}^{\mathcal{B}}, \mathbf{D})$$

□

We will introduce one last property and prove that TBIC holds this property. This property, together with Proposition 5, is believed to be important for conducting a future possible proof of sound and completeness of TGES.

Definition 10. *(Locally consistent scoring criterion according to tiered background knowledge $\mathcal{K}$) Let* $\mathbf{D}$ *be a set of data consisting of m records that are i.i.d. samples from some distribution p. Let $\mathcal{G}$ be any DAG, and let $\mathcal{G}'$ be the DAG that results from adding the edge $X_i \rightarrow X_j$. A scoring criterion $S(\mathcal{G}, \mathbf{D})$ is locally consistent according to tiered background knowledge $\mathcal{K}$ if the following properties hold:*

1. *If $\mathcal{G}$ encodes $\mathcal{K}$, $\tau(X_i) \leq \tau(X_j)$ and $X_j \not\perp\!\!\!\perp_p X_i \mid Pa_j^{\mathcal{G}}$, then $S(\mathcal{G}', \mathbf{D}) > S(\mathcal{G}, \mathbf{D})$*
2. *If $\mathcal{G}$ encodes $\mathcal{K}$, $\tau(X_i) \leq \tau(X_j)$ and $X_j \perp\!\!\!\perp_p X_i \mid Pa_j^{\mathcal{G}}$, then $S(\mathcal{G}', \mathbf{D}) < S(\mathcal{G}, \mathbf{D})$*
3. *If $\mathcal{G}$ encodes $\mathcal{K}$, $\tau(X_i) > \tau(X_j)$ then $S(\mathcal{G}', \mathbf{D}) < S(\mathcal{G}, \mathbf{D})$*

Proposition 6. *TBIC is a locally consistent scoring criterion according to tiered background knowledge* $\mathcal{K}$

Proof. 1 & 2: If $\mathcal{G}$ and $\mathcal{G}'$ encodes the tiered background knowledge $\mathcal{K}$, then the scoring criterion TBIC is equal to the scoring criterion *BIC* which by (D. M. Chickering, 2003) has property 1 and 2.
3: Assume $\mathcal{G}$ encodes $\mathcal{K}$ and $\tau(X_i) > \tau(X_j)$ then $\mathcal{G}'$ contradicts $\mathcal{K}$ and the property follows from the fact that $\tilde{S}(\mathcal{G}', \mathbf{D}, \mathcal{K}) = S_B(\mathcal{G}', \mathbf{D}) > -\infty$ and $\tilde{S}(\mathcal{G}, \mathbf{D}, \mathcal{K}) = -\infty$. □

3.4 TGES

In this section, we will use the newly defined score criterion TBIC to develop a new method which avoids the issue of Simple TGES described in Remark 1. This new method will also incorporate tiered background knowledge into a Greedy Equivalence Search algorithm, but it will use the tiered background knowledge earlier in the process in the pursuit of better performance outside of the oracle scenario. We will start out by introducing the algorithm *Temporal Greedy Equivalence Search* (TGES), and then we will prove some properties for it. Afterwards, we will present the implementation and some of the thoughts and work behind it. In the last subsection, we will present an example using TGES, Simple TGES, and GES.

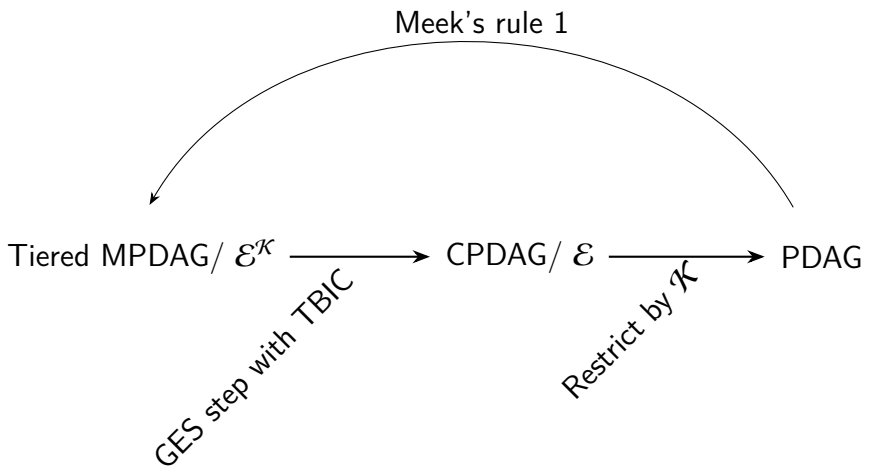


Figure 6: Illustration on how TGES moves between different types of graphs in one step. TGES runs through 3 phases, and each phase consists of steps until optimum is reached. Each step start and stops at "Tiered MPDAG/$\mathcal{E}^{\mathcal{K}}$

TGES (Temporal Greedy Equivalence Search) has 3 phases that are analogous to GES: forward, backward, and turning. TGES starts off with an empty graph like in the original GES. The forward phase adds edges according to TBIC step by step in a greedy way. The edge added is the edge that adds the biggest increase in the score criterion. Each step then finds the CPDAG/$\mathcal{E}$ of the graph with the newly added edge, then the CPDAG is restricted according to $\mathcal{K}$, and we use Meek's rule 1 (See Appendix A.2) (Bang & Didelez, 2023) to infer extra orientations of edges. The resulting graph after a full iteration of a step is a tiered MPDAG. Step by step, we add edges until a (possibly local) maximum is reached. In Figure 6, the operations in one step of TGES are illustrated.

After the forward phase reaches a maximum, TGES initiates a backward phase. The steps work in the same way as the forward phase but optimises by removing edges (See Figure 6). The backward steps continue until no improvement can be made to the score by removing an edge and we have reached a maximum.

The turning phase then commences in the same way as the two others. When the turning phase reaches a maximum, the forward phase starts over. We keep iterating through the three phases until no edges are added, removed, or turned.

3.4.1 Properties of TGES

We will, in this subsection, prove that TGES ensures that our prediction will be a tiered MPDAG, even when not in the oracle scenario.

First, note that a forward, backward, or turning step in the TGES algorithm corresponds to first running the corresponding step from the GES algorithm with score function $\tilde{S}$ on a tiered MPDAG. GES, by design, then terminates with a CPDAG. This new CPDAG is then restricted using the tiered background knowledge $\mathcal{K}$ and Meek's rule 1 to obtain a tiered MPDAG. To ensure that we in fact find a tiered MPDAG, we need to make sure that the tiered background knowledge $\mathcal{K}$ is in agreement with the CPDAG obtained from the GES step with $\tilde{S}$.

Lemma 3. *The GES forward step with score function $\tilde{S}$ applied to a restricted equivalence class $\mathcal{E}^{\mathcal{K}}$ terminates in a CPDAG that is in agreement with the tiered background knowledge $\mathcal{K}$.*

Proof. We have that all DAGs $\mathcal{G} \in \mathcal{E}^{\mathcal{K}}$ encode the tiered background knowledge. By design, all of these DAGs have that their score $\tilde{S}(\mathcal{G}, \mathbf{D}, \mathcal{K}) > -\infty$.
Assume that the step terminates in an equivalence class $\mathcal{E}'$. Assume that $\mathcal{G}^* \in \mathcal{E}^{\mathcal{K}}$ is the specific DAG that, by adding an edge to, becomes a DAG $\mathcal{G}'$ in $\mathcal{E}'$. By definition of $\boldsymbol{\mathcal{E}}^+(\mathcal{E}^{\mathcal{K}})$, both of these DAGs exist.
Assume for contradiction that the equivalence class $\mathcal{E}'$ is not in agreement with $\mathcal{K}$.
Therefore, all DAGs $\mathcal{G} \in \mathcal{E}'$ contain an edge that contradicts $\mathcal{K}$, and by design of the score, we have that for all DAGs $\mathcal{G} \in \mathcal{E}'$ that $\tilde{S}(\mathcal{G}, \mathbf{D}, \mathcal{K}) = -\infty$. This also means that $\tilde{S}(\mathcal{G}', \mathbf{D}, \mathcal{K}) = -\infty$. But since GES is greedy and $\tilde{S}(\mathcal{G}^*, \mathbf{D}, \mathcal{K}) > \tilde{S}(\mathcal{G}', \mathbf{D}, \mathcal{K})$ there is a contradiction. □

Lemma 4. *The GES backward step with score function $\tilde{S}$ applied to a restricted equivalence class $\mathcal{E}^{\mathcal{K}}$ terminates in a CPDAG that is in agreement with the tiered background knowledge $\mathcal{K}$.*

Proof. For any $\mathcal{G}^* \in \mathcal{E}^{\mathcal{K}}$, we have that it contains no edges that contradict $\mathcal{K}$. Since $\mathcal{K}$ only restricts which edges cannot be there and does not require any edges to remain, any edge removal from $\mathcal{G}^*$ resulting in a new DAG $\mathcal{G}'$ will also not contradict $\mathcal{K}$. Therefore, there exists at least one DAG in $\mathcal{E}(\mathcal{G}')$ that does not contradict $\mathcal{K}$, and we have that $\mathcal{E}(\mathcal{G}')$ and the corresponding CPDAG is in agreement with $\mathcal{K}$. □

Lemma 5. *The GES turning step with score function $\tilde{S}$ applied to a restricted equivalence class $\mathcal{E}^{\mathcal{K}}$ terminates in a CPDAG that is in agreement with the tiered background knowledge $\mathcal{K}$.*

Proof. Note that this proof is analogous to the proof of Lemma 3
We have that all DAGs $\mathcal{G} \in \mathcal{E}^{\mathcal{K}}$ encode the tiered background knowledge. By design, all of these

DAGs have that their score $\tilde{S}(\mathcal{G}, \mathbf{D}, \mathcal{K}) > -\infty$.
Assume that the step terminates in an equivalence class $\mathcal{E}'$. Assume that $\mathcal{G}^* \in \mathcal{E}^{\mathcal{K}}$ is the specific DAG that by reversing an edge to, becomes a DAG $\mathcal{G}'$ in $\mathcal{E}'$. By definition of $\boldsymbol{\mathcal{E}}^{\leftrightarrow}(\mathcal{E}^{\mathcal{K}})$ both of these DAGs exist.
Assume for contradiction that the equivalence class $\mathcal{E}'$ is not in agreement with $\mathcal{K}$.
Therefore all DAGs $\mathcal{G} \in \mathcal{E}'$ contains an edge that contradicts $\mathcal{K}$, and by design of the score we have that for all DAGs $\mathcal{G} \in \mathcal{E}'$ that $\tilde{S}(\mathcal{G}, \mathbf{D}, \mathcal{K}) = -\infty$. This also means that $\tilde{S}(\mathcal{G}', \mathbf{D}, \mathcal{K}) = -\infty$. But since GES is greedy and $\tilde{S}(\mathcal{G}^*, \mathbf{D}, \mathcal{K}) > \tilde{S}(\mathcal{G}', \mathbf{D}, \mathcal{K})$ there is a contradiction. □

The collection of Lemmas 3 to 5 ensures that our prediction will always be a tiered MPDAG.

Lemmas 3 to 5 could be extended to also take in the case of non tiered background knowledge, by substituting the score function for score function that takes all types of background knowledge into account. The proof for backward step is in this case proved in similar manner to forward and turning.

We introduce the following two conjectures that together result in TGES being optimal in the limit of large sample size. The rationale of the conjectures being true, stems from the properties of TBIC, local consistency (Proposition 6) and optimal in the graph that is a perfect map of *p* (Proposition 5). In Section 5, we will see a simulation study that also backs up these conjectures being true. Ideas for how to prove these conjectures are discussed later in Section 6.

Conjecture 1. *(Tiered optimality of TGES forward phase)*
Let $\mathcal{E}^{\mathcal{K}}$ be the restricted equivalence class that is a result from doing a forward phase in TGES, let p be the distribution from which the data **D** *was generated, m be the number of observations in* **D** *and $\mathcal{K}$ the tiered background knowledge of p. Then in the limit of large m, p is contained in $\mathcal{E}^{\mathcal{K}}$, and all DAGs in $\mathcal{E}^{\mathcal{K}}$ encode $\mathcal{K}$.*

Conjecture 2. *(Tiered optimality of TGES backward phase)*
Let $\mathcal{E}^{\mathcal{K}}$ be the restricted equivalence class that is a result from doing a backward phase in TGES starting out with a restricted equivalence class containing p and where all DAGs inside the class encode $\mathcal{K}$. Here p is the distribution from which the data **D** *was generated, m the number of observations in* **D** *and $\mathcal{K}$ the tiered background knowledge of p. Then in the limit of large m, $\mathcal{E}^{\mathcal{K}}$ is a perfect map of p and all DAGs in $\mathcal{E}^{\mathcal{K}}$ encode $\mathcal{K}$.*

3.4.2 Implementation

This section seeks to describe the implementation of TGES, the problems that arose when trying to implement it and how these problems where dealt with. We will also give an overview of the implementation by introducing a pseudo code for TGES.
The implementation of TGES has been made in R and requires the pcalg R package (Kalisch et al., 2024). The code for it can be found in the script ”TGES_function.R” (Larsen, 2024).

The ”tges()” function takes in a class defining the score, data and a tiered ordering. The function is then made up of one big while loop checking if all three phases did not modify any edges, and three smaller while loops checking whether a phase reached a local maximum. While we in theory only need to use Meek’s rule 1 to obtain a tiered CPDAG (Bang & Didelez, 2023) we in the code check for all 4 of Meek’s rules (See Appendix A.2 for Meek’s rules). The function ”tges()” can also print out modified edges, score differences that result in a modified edge, and which of Meek’s rules were used to infer directions (this is always Meek’s rule 1 as insured by Lemmas 3 to 5).

To be able to implement a viable TGES by using GES it is also important to understand how the GES from pcalg (Kalisch et al., 2024) adds edges to a graph $\mathcal{G}$. It scores edges locally by computing the loglikehood of the Maximum Likelihood Estimate (MLE) for linear regression on a node X_i with its $Pa_i^{\mathcal{G}}$ as covariates and penalizes with $\frac{2+\#\text{parents}}{2}\log(m)$. Then, it compares the scores of similar models with an extra possible node added to the set of parents. If any of the new scores are higher, it adds the new directed edge. If the edge cannot be directed in the CPDAG of the graph, then an undirected edge is added, which in the code corresponds to adding a directed edge in both directions. The backward phase will remove an undirected edge if it can obtain a better score by removing the edge in either direction.

Remark 2. *Assume we have an undirected edge X —Y of a graph (CPDAG or tiered MPDAG) and assume that there are two DAGs $\mathcal{G}, \mathcal{G}'$ contained in the graph, where the edge $X \to Y$ is in $\mathcal{G}$ and $X \leftarrow Y$ is in $\mathcal{G}'$.*
If our score S is not score equivalent in the class (Markov equivalence class or restricted equivalence class) such that $S(\mathcal{G}) \neq S(\mathcal{G}')$ and we are scoring both $\mathcal{G}$ and $\mathcal{G}'$ in the forward and backward phase then the greedy search algorithm might run in an infinite loop adding and removing the undirected edge X —Y.

Remark 2 motivated the implementation of TGES and Simple TGES.
Provided below in Algorithm 3 is the pseudo code for TGES. The score referred to is TBIC.

Algorithm 3 Temporal Greedy Equivalence Search (TGES)

Initialize:

$\mathcal{E}^{\mathcal{K}} \leftarrow$ the restricted equivalence class* of the empty DAG (no edges).
Compute initial score $\tilde{S}(\mathcal{E}^{\mathcal{K}}, \mathbf{D}, \mathcal{K})$.
while there are changes that improve the score **do**
    Forward phase
    for all non-adjacent nodes (X, Y) in $\mathcal{E}^{\mathcal{K}}$ **do**
        $\hat{\mathcal{E}}^{\mathcal{K}} \leftarrow \mathcal{E}^{\mathcal{K}}$ with added directed edge from X to Y
        $\mathcal{G}' \in \hat{\mathcal{E}}^{\mathcal{K}}$
        Compute $\tilde{S}(\mathcal{G}', \mathbf{D}, \mathcal{K})$.
    end for
    if any score of $\mathcal{G}'$ is bigger **then**
        $\mathcal{G} \leftarrow \underset{\mathcal{G}'}{\operatorname{argmax}} \left(\tilde{S}(\mathcal{G}', \mathbf{D}, \mathcal{K})\right)$
        $\mathcal{E}^{\mathcal{K}} \leftarrow (\mathcal{E}(\mathcal{G})$ restricted according to $\mathcal{K}$ and Meek's rule 1)
    end if
    Backward phase
    for all adjacent nodes (X, Y) in $\mathcal{E}^{\mathcal{K}}$ **do**
        $\hat{\mathcal{E}}^{\mathcal{K}} \leftarrow \mathcal{E}^{\mathcal{K}}$ with removed directed edge from X to Y
        $\mathcal{G}' \in \hat{\mathcal{E}}^{\mathcal{K}}$
        Compute $\tilde{S}(\mathcal{G}', \mathbf{D}, \mathcal{K})$.
    end for
    if any score of $\mathcal{G}'$ is bigger **then**
        $\mathcal{G} \leftarrow \underset{\mathcal{G}'}{\operatorname{argmax}} \left(\tilde{S}(\mathcal{G}', \mathbf{D}, \mathcal{K})\right)$
        $\mathcal{E}^{\mathcal{K}} \leftarrow (\mathcal{E}(\mathcal{G})$ restricted according to $\mathcal{K}$ and Meek's rule 1)
    end if
    Turning phase
    for all directed edges between nodes (X, Y) in $\mathcal{E}^{\mathcal{K}}$ **do**
        $\hat{\mathcal{E}}^{\mathcal{K}} \leftarrow \mathcal{E}^{\mathcal{K}}$ with reversed directed edge from X to Y
        $\mathcal{G}' \in \hat{\mathcal{E}}^{\mathcal{K}}$
        Compute $\tilde{S}(\mathcal{G}', \mathbf{D}, \mathcal{K})$.
    end for
    if any score of $\mathcal{G}'$ is bigger **then**
        $\mathcal{G} \leftarrow \underset{\mathcal{G}'}{\operatorname{argmax}} \left(\tilde{S}(\mathcal{G}', \mathbf{D}, \mathcal{K})\right)$
        $\mathcal{E}^{\mathcal{K}} \leftarrow (\mathcal{E}(\mathcal{G})$ restricted according to $\mathcal{K}$ and Meek's rule 1)
    end if
end while
Output: $\mathcal{E}^{\mathcal{K}}$

*: The class of empty DAGs is a restricted equivalence class and is consistent with all tiered background knowledge since tiered background knowledge only states which directed edges are forbidden. Also, all edges (which are none) are maximally directed.

3.4.3 Example

Code for reproducing the example can be found in the script "Example of TGES.R" (Larsen, 2024).

We simulate data from a Gaussian distribution with dependency structure according to an underlying data-generating DAG like the one in Figure 7 with tiers $\tau(A) = 1, \tau(B) = 2, \tau(C) = \tau(D) = \tau(E) = 3$. We simulate a data set of sample size 10000 and fit GES from pcalg (Kalisch et al., 2024), TPC with Gaussian independence test and sparsity level 0.01 from tpc (Witte, 2023), TGES, and Simple TGES on the data.
The structural equations for the simulation are given as:

$$
\begin{aligned}
A &= \epsilon_A \\
B &= 0.623A + \epsilon_B \\
D &= 0.828B + \epsilon_D \\
E &= 0.854B + 0.978A + \epsilon_E \\
C &= 0.521A + 0.997B + 0.727E + \epsilon_C
\end{aligned}
$$

Where $\epsilon_A, \epsilon_B, \epsilon_C, \epsilon_D, \epsilon_E$ are i.i.d standard normally distributed.

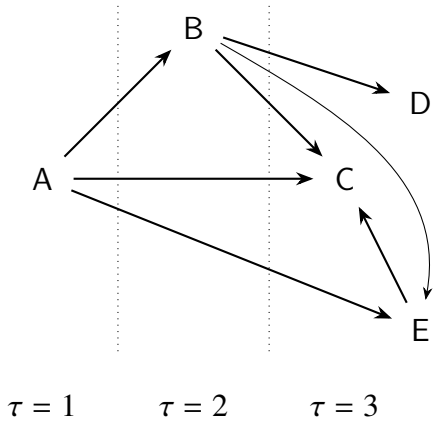


Figure 7: Underlying data-generating DAG

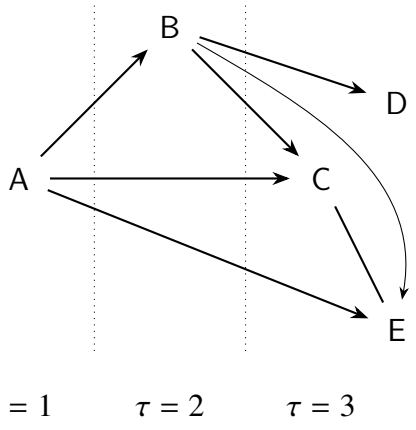


Figure 8: True tiered MPDAG. Predicted by TGES and TPC

Figure 7 is the true data-generating DAG, and Figures 8 to 10 are all predicted graphs by the algorithms.
Note that the edges of true data-generating DAG Figure 7 consist of 6 out of 7 cross-tier edges and that there are no v-structures.

We see that TPC and TGES both predict a tiered MPDAG that contains the underlying data-generating DAG as illustrated in Figure 8. We have that GES predicts a CPDAG like the one represented by Figure 9, which almost has the same skeleton as the true graph but has very different orientations of the adjacencies. The Simple TGES estimates the graph represented by Figure 10, which has 3 out of 4 correct adjacencies, and 2 out of 3 of those are oriented correctly.

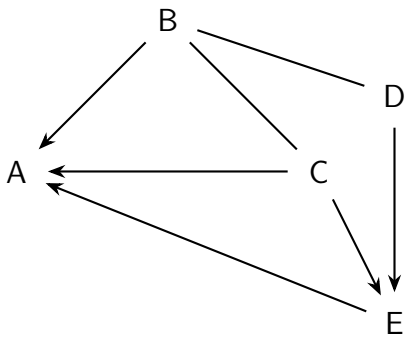


Figure 9: CPDAG predicted by GES

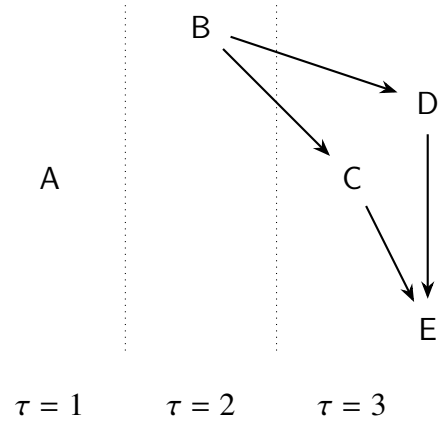


Figure 10: Tiered[Remark 1] MPDAG predicted by Simple TGES

We see that even with a sample size as big as 10000 and only 5 nodes, Simple TGES is not able to correctly predict the restricted equivalence class like it is in the limit of large sample size.

4 Methods for simulation and evaluation

We wish to assess the performance of the methods we have created outside of the theoretical limit in large sample size. We especially want to compare them to earlier work in causal discovery with tiered background knowledge, and compare them to Greedy Equivalence Search without tiered background knowledge. To do this assessment of methods we will perform a simulation study.

This section introduces the functions, methods, and metrics used to perform the simulation study introduced later in Section 5. All these are necessary tools to be able to perform the simulation study we wish to do on the performance of our new algorithms. As to our knowledge, no earlier work has performed a similar test on preexisting causal discovery algorithms with tiered background knowledge. Hence, we have developed a way of simulating a DAG with predefined tiered background knowledge, and we have created a new metric that will be defined in this section.
We will start by introducing the implementation of generating a random tiered DAG with jointly Gaussian data in Section 4.1. Next, we will introduce metrics used for evaluating a predicted tiered MPDAG in Section 4.2.

4.1 Simulating a data-generating DAG with background knowledge

We assumed there to be a true data-generating DAG with corresponding true jointly Gaussian data-generating distribution p.

We will in this subsection introduce two R functions, *randtierDAG* and *rmvTDAG*, that play a role in generating a random tiered DAG with jointly Gaussian data, both functions can be found in the script ”Simulate_functions.R” (Larsen, 2024).

The aim of generating a DAG is to check the performance of our new causal discovery method TGES, hence the function creating the data-generating DAG takes in a set of arguments believed to have influence on the performance of the algorithms.
The function which generates a DAG with tiers is called *randtierDAG* and takes in 5 different arguments: **incpar**, **accpar**, **tino**, **lB** and **uB**.

incpar is the probability of having an edge between two arbitrary nodes in the same tier. Hence, an edge $\{A \rightarrow B\}$ between two nodes A and B with ordering τ where $\tau(A) = \tau(B)$ exist with probability **incpar** independently from all other edges. We have that **incpar** $\in [0, 1]$. The directions of the edges are chosen from a topological order of the nodes, which is drawn uniformly from the set of permutations of the in-tier nodes.

accpar is the probability parameter of having a cross-tier edge between two nodes. Hence, an edge $\{A \rightarrow B\}$ between two nodes A and B with ordering τ where $\tau(A) < \tau(B)$ exist with probability **accpar** independently from all other edges. We have **acccpar** $\in [0, 1]$.

tino is a vector that stores the number of nodes in each tier. For example, $(3, 4)$ indicates 3 nodes in the first tier and 4 nodes in the second tier. The number of nodes and the number of tiers are implied by **tino** since *length*(**tino**) = #tiers and *sum*(**tino**) = #nodes.

lB and **uB** are the lower and upper bound of the *weights* of an edge between two nodes. Given the existence of an edge, the coefficient is drawn from a uniform distribution between **lB** and **uB**. For now the function only takes in positive values for both **lB** and **uB**, and clearly **lB** ≤ **uB**.

The ability to be able to differentiate between graphs with different sparsity of cross-tier edges and in-tier edges, provides an opportunity to differentiate between two types of sparsity of edges in the data-generating DAG and whether they have different effect on the causal discovery methods.
In practice the DAG encoding the background knowledge given by **tino** is generated by generating a sub-DAG for each each tier with the *r.gauss.pardag* function from the *pcalg* package(Kalisch et al., 2024) with values p = #nodes in the tier, prob = **incpar**, lbe = **lB** and ube = **uB**. *r.gauss.pardag* generates a DAG by drawing an undirected graph from an Erdős-Rényi model[4] and then orienting edges according to a random topological ordering (Kalisch et al., 2024). After generating the sub-DAGs, we insert the cross-tier edges with probability **accpar** and weights drawn randomly from a uniform distribution with min = **lB** and max = **uB**. The ordering of the tiers ensures that no cycles are created in the process of adding cross-tier edges.
The output of the function *randtierDAG* is a *weighted adjacency matrix* W_m of the generated random DAG with at least one non-zero entry.
For further description on how a DAG is simulated see the pseudo code, Algorithm 4, provided below.

[4]Model where each edge has a probability of existing independently of the other edges.

Algorithm 4 Function randtierDAG

Require:

incpar $\in [0, 1]$: Probability of in-tier edges

accpar $\in [0, 1]$: Probability of cross-tier edges

tino: Vector of #nodes per tier

lB, **uB**: Lower and upper bounds of edge weights

$0 \leq$ **lB** $\leq$ **uB**

Initialize:

1: #tiers $\leftarrow$ length(tino)
2: #nodes $\leftarrow$ sum(tino)
3: **for** each tier **do**
4: Generate in-tier sub-DAG $\mathcal{G}'$:
5: #nodes according to tino
6: Weight of causal effects $\leftarrow U(lB, uB)$
7: Probability of edge $\leftarrow$ incpar
8: **end for**
9: $\mathcal{G} \leftarrow \bigcup_{\text{each tier}} \mathcal{G}'$
10: **for** each pair of nodes (X_i, X_j) in $\mathcal{G}$ with $\tau(X_i) < \tau(X_j)$ **do**
11: Insert edge with probability accpar
12: **if** edge inserted **then**
13: Weight of causal effect $\leftarrow U(lB, uB)$
14: **end if**
15: **end for**
16: **if** $\mathcal{G}$ is empty **then**
17: Go to line 3
18: **end if**
19: **Output:** $\mathcal{G}$

When we have obtained a weighted adjacency matrix, we can simulate observations from a jointly Gaussian distribution according to the weighted adjacency matrix with the function *rmvTDAG* (Larsen, 2024). The function *rmvTDAG* is given a weighted adjacency matrix W_m and a sample size m.

We start by ordering the d nodes by their topological ordering. For easier notation, we denote the first node in the ordering as X_1, the next as X_2, and so on. We then draw d exogenous i.i.d. multivariate standard normal distributions, one assigned for each node. We denote them ϵ_i for

$i \in \{1, .., d\}$. Then we set $X_1 = \epsilon_1$. After this we for every $i \in \{1, .., d\}$ we set

$$X_i = \begin{pmatrix} X_1 & ... & X_{i-1} \end{pmatrix} \begin{pmatrix} w_{1,j} \\ \vdots \\ w_{i-1,j} \end{pmatrix} + \epsilon_i$$

The topological ordering makes sure a node is defined only by nodes that have been defined before it.

4.2 Performance metrics

The true data-generating DAG is typically not possible to fully identify, as our methods find restricted equivalence classes or equivalence classes and not specific DAGs. Therefore, we will compare our predictions to the restricted equivalence class of the true data-generating DAG or, more specifically, its corresponding tiered MPDAG. We will refer to this tiered MPDAG as the true tiered MPDAG.
In this subsection, we will give an introduction to the two metrics, precision and recall of a tiered MPDAG and a CPDAG. These metrics may be used to quantify how well a method performs. Each metric measures how much information is captured by the predicted restricted equivalence class compared to the restricted equivalence class of the true data-generating DAG.
We will begin by introducing the concept of a confusion matrix and give three ways of computing its entries. From the definition of a confusion matrix, we will define the metrics precision and recall.

We start by introducing the concept of a *confusion matrix*, which holds the building blocks of the metrics we wish to use. It is defined from a dichotomous prediction of conditions and the true dichotomous conditions. The confusion matrix consist of 4 counts: *True Positives (TP)*, *False Positives (FP)*, *True Negatives (TN)* and *False Negatives (FN)*. Each count is the number of conditions in the respective combination of either true positive prediction, false positive prediction, etc. The definition of what constitutes a positive and a negative, will be described later on.

We will define the confusion matrix from the tiered MPDAG rather than the restrictive equivalence class. A tiered MPDAG can represent a restricted equivalence class (Bang & Didelez, 2023), hence we may refer to a confusion matrix of the restricted equivalence class, but actually mean the confusion matrix of the corresponding tiered MPDAG. The definitions of what constitutes a TP, FP, TN, and FN, with respect to a tiered MPDAG, depend on what part of the graph's structure we wish to measure. We will give a definition that is equivalent over any type of PDAGs, here

among tiered MPDAG and CPDAG. We will introduce three different approaches to defining what constitutes a TP, FP, TN, and FN, concentrating on either *adjacencies*, *directions* (Petersen, 2022) or *in-tier directions*. The metric concentrating on in-tier directions is a metric we have created to enhance comparability on directions of a CPDAG and a tiered MPDAG.

4.2.1 Confusion matrix on adjacencies

When defining the values of the confusion matrix from adjacencies, we will not differentiate between directed or undirected edges and will only be interested in whether there is an adjacency between two nodes or not. Hence, a TP would mean that the predicted graph and the true graph agree on the presence of an adjacency, a TN means both graphs agree on the absence of an adjacency. An FP is whenever the predicted graph has an adjacency that is not present in the true graph, and an FN is whenever there is an absence of an adjacency in the predicted graph which is present in the true graph. Other literature, such as (J. D. Ramsey & Andrews, 2017) and (Petersen, 2022), has used the same definition of entries in a confusion matrix of adjacencies, and the implementation of this in R has been done by (Petersen, 2022). An overview on how different edges between two nodes are measured according to adjacencies is located in Table 11 in Appendix A.3.

4.2.2 Confusion matrix on directions

The second approach for defining the values of the confusion matrix concentrates on directions of edges. We look at the agreement of a direction between the predicted graph and the true graph given that they agree on an adjacency between two nodes. A TP would mean that the predicted and true graph agree on the direction of the edge, a TN means that the predicted and true graph agree that the edge is undirected, an FP means that the true edge is undirected while the predicted edge is directed or that the true and predicted graph disagree on the direction of the edge, and an FN means that the true edge is directed while the predicted edge is either wrongly directed or undirected. While the concept of focusing on directions when measuring the amount of information captured by a predicted graph is broadly used in causal discovery, there is not always agreement on exactly how to define it. In some literature like (J. D. Ramsey & Andrews, 2017) it is not entirely obvious how this metric is defined, while in other literature there is a concrete difference in the definition (see fig 4 in Kummerfeld et al., 2024 where they base the metric on arrowheads instead). The definition used in this thesis is a variation of the one implemented in the R package causalDisco

where a small bug fix has been fixed[5] (Petersen, 2022). An overview of how different edges are measured according to directions can be found in Table 2.

Estimated edge	True edge	Result
A —— B	A —— B	TN
A —— B	A ⟶ B	FN
A ⟶ B	A —— B	FP
A ⟶ B	A ⟶ B	TP
A ⟶ B	A ⟵ B	FP, FN
A B	A any edge B	0
A any edge B	A B	0

Table 2: This table indicates how the elements of the confusion matrix of directions are computed from a graph. 0 indicates no effect on the elements of the confusion matrix of directions.

4.2.3 Confusion matrix on in-tier directions

This metric has been created with the purpose of being able to evaluate what directional information is gained by having the tiered background knowledge, other than the trivial orientations given directly by the tiered background knowledge. We know that both TGES and Simple TGES orient extra edges using Meek's rule 1, but we wish to measure this ability and compare it between methods. This measure gives opportunity for measuring the possible non-trivial directional information gained by using a method that takes tiered background knowledge into account. This gives rise to the idea of defining a new type of confusion matrix on directions that only focuses on edges between two nodes in the same tier. This will be a confusion matrix with entries defined on the values of in-tier directions and hence also depends on the tiered background knowledge. The metric is defined as the matrix addition of the confusion matrix of directions of each in-tier sub-DAG. An in-tier sub-DAG is the sub-DAG of all nodes and edges in the same tier. This thesis is, to our knowledge, the first to use this type of metric and the metric has been implemented in R as a function *intier_confusion* which can be found in "Simulate_functions.R" Larsen (2024). An overview

[5]See "Simulate_functions.R" in (Larsen, 2024)

of how different edges are measured according to in-tier directions can be found in Table 3.

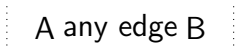
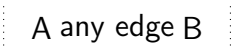
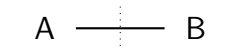
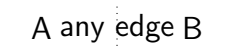
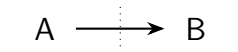
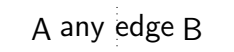
Estimated edge	True edge	Result
		Same as Table 2
		0
		0

Table 3: This table indicates how the elements of the confusion matrix of in-tier directions are computed from a graph. 0 indicates no effect on the elements of the confusion matrix of in-tier directions and the indexed *t*'s indicates tiers with a dashed line separating two tiers.

4.2.4 Metrics

We will define the two metrics, precision and recall, solely on the entries of the confusion matrix. The metrics are used by other authors, such as Hasan and Gani (2023), Raghu et al. (2018) and J. D. Ramsey and Andrews (2017), that wish to evaluate the performance of causal discovery algorithms. There exist several other metrics, but in the interest of evaluating in the simplest way that is still expressive of the performance of the algorithm, only precision and recall are reported in this thesis.

The first metric is called *precision* and is defined as $\frac{TP}{TP+FP}$. Precision is a measure of the proportion of correct hits among all positive predictions. The interpretation of precision is similar between all three kinds of confusion matrices, but there are some differences:

Precision of adjacencies: How big of a proportion of the predicted adjacencies are actually true. It is a measure on how precise the predicted skeleton of the graph is. This metric tends to punish dense graphs and reward sparser graphs.

Precision of directions: The amount of correctly directed edges relative to the amount of edges wrongly oriented and the amount of edges that are predicted as directed even though the true restrictive equivalence class does not allow for orienting the edge. This metric punishes orienting

edges too much and orienting edges wrongly. Predicting a directed edge as undirected affect this metric in neither a negative or positive way.

Precision of in-tier directions: This metric is equal to the precision of directions, but only for the edges that are between two nodes in the same tier.

The second metric *recall* is defined as $\frac{TP}{TP+FN}$. Recall is a measure of the proportions of correct hits among all potential hits. The interpretations of the three introduced definitions of entries of the confusion matrix are as follows:

Recall of adjacencies: The proportion of the true adjacencies that have been predicted. This is a measure on the ability of the prediction to detect the true skeleton. This metric tends to punish sparse graphs and reward denser graphs.

Recall of directions: This is the amount of correctly predicted directions relative to the amount of wrongly directed edges and the amount of predicted undirected edges that according to the true restricted equivalence class could be directed. This metric punishes directing edges wrongly just like precision of directions, but also punishes too ”vague” edges, here meaning undirected edges that, according to the true restrictive equivalence class, could be directed.

Recall of in-tier directions: This metric is equal to the recall of directions, but only for the edges that are between two nodes in the same tier.

Neither of the two measures takes True Negatives into account, which could be equally as important as precision and recall. As mentioned before, the reasoning for only including recall and precision has been to limit the number of different metrics on which to evaluate the performance of the predictions and the fact that recall and precision have been used by other authors to evaluate causal discovery algorithms (see Hasan & Gani, 2023, Raghu et al., 2018 and J. D. Ramsey & Andrews, 2017). The choice to narrow it down to two metrics has been made to ensure that the findings would be comprehensible, but further research might find it relevant to incorporate TN.

5 Simulation study

In this section, we wish to measure the performance of the TGES compared to other causal discovery methods that take tiered background knowledge into account. We furthermore quantify the benefits and drawbacks gained by adding tiered background knowledge to a score-based approach such as GES. We will evaluate the methods with the metrics introduced in Section 4 and report relevant representations.
We will introduce a three part simulation study, each part with a different focus. Section 5.1 gives a general overview of the performance of the relevant causal discovery methods and compares TGES with alternative causal discovery methods with tiered background knowledge. Section 5.1 also compares TGES to the use of no tiered background knowledge and evaluates the up- and downside of utilizing tiered background knowledge. Section 5.2 measures the relevant methods across different sets of parameters to explore whether a change in parameters yields a change in the prediction power of the methods. Section 5.3 addresses the computational times of the different methods.

5.1 Performance of methods in a general setting

The simulations in this subsection can be reproduced with the script ”Simulation study boxplots.Rmd” (Larsen, 2024).

In this first part of the simulation study, the aim is to give a more general overview of the methods of relevance. We randomly draw the different parameters used for *randtierDAG* and generate datasets on which we can evaluate our methods. The parameters are chosen uniformly as follows:

incpar $\in (0, 1)$

accpar $\in (0, 1)$

#nodes $\in \{7, 8, ..., 19, 20\}$

#tiers $\in \{2, 3, 4, 5\}$

tino is chosen as a random vector of length equal to #tiers, *sum*(**tino**) = #nodes and no 0 entries.

We fix the sample size to $m = 10000$, to ensure having enough data to evaluate the methods instead of evaluating the lack of data.
The possible weights of the weighted adjacency matrix have a lower bound (**lB**) of 0 and an upper

bound (**uB**) of 1. The weights are drawn randomly, like described in Section 4.1. The data is drawn from a multivariate Gaussian distribution with dependency structure according to the weighted adjacency matrix.
We evaluate on 800 different sets, with each set containing data and the corresponding underlying data-generating DAG. Each method is fitted on the same data and tiered background knowledge.

The methods evaluated in this section are: TGES, Simple TGES, GES, and TPC with a Gaussian independence test and sparsity level 0.1 (we also evaluated TPC with sparsity level equal to 0.01 and 0.001, see Appendix A.4, Figure 18).

The metrics used in this section are precision and recall on adjacencies, directions and in-tier directions. Each metric compares either the predicted tiered MPDAG (TGES, Simple TGES and TPC) or the predicted CPDAG (GES) to the true tiered MPDAG. The true tiered MPDAG being the tiered MPDAG containing the underlying data-generating DAG.

5.1.1 Adjacencies and directions

We start by comparing the TGES method to the two other methods that take background knowledge into account: TPC and Simple TGES. We will compare to TPC with sparsity level equal to 0.1 since TPC with lower sparsity levels exhibit same relative structure with respect to TGES, and TGES is the main focus of this thesis. We do however see that lower sparsity level results in higher precision of adjacencies and lower recall of adjacencies for TPC, as can be seen from Figure 18 in Appendix A.4.

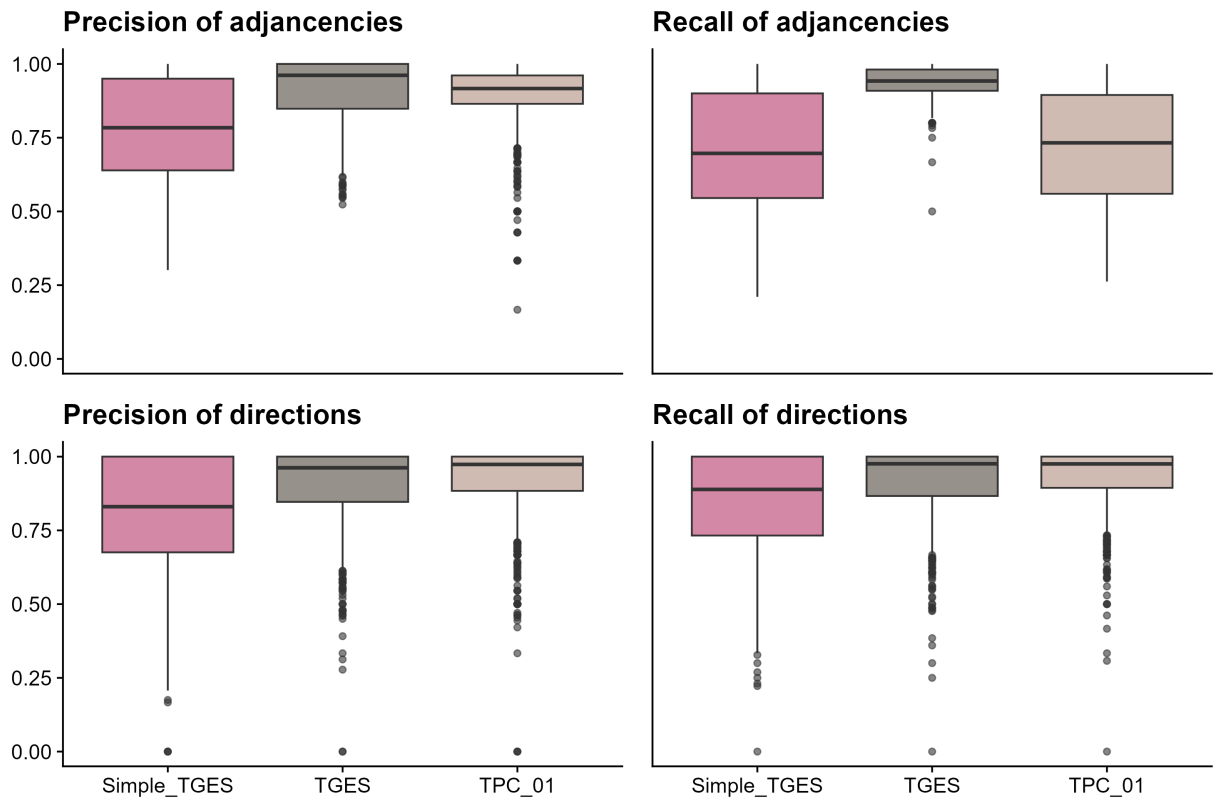


Figure 11: Precision and recall of adjacencies and directions computed for 800 simulations. The methods included are Simple TGES (red), TGES (black), and TPC with sparsity level 0.1 (light brown).

We see from Figure 11 that TGES on average performs better than Simple TGES on both directions and adjacencies. The adjacency metrics show us that TGES does a better job than Simple TGES at predicting the skeleton. This suggests that using the tiered background knowledge "earlier" in the algorithm helps improve the estimation of the skeleton. Even though the tiered background knowledge does not restrict on the adjacencies directly, how we utilize the tiered background knowledge has an effect on the prediction of the skeleton. By ensuring that the steps in the score-based greedy approach stay within restrictive equivalence classes $\mathcal{E}^{\mathcal{K}}$ in agreement with background knowledge $\mathcal{K}$ (TGES), we estimate the skeleton with higher precision and recall than if we would only restrict the equivalence class $\mathcal{E}$ using the tiered background knowledge $\mathcal{K}$ as the last step (Simple TGES).

The poorer performance of adjacency metrics of Simple TGES could be accredited to the algorithm in some instances restricting an equivalence class $\mathcal{E}$ that is not in agreement with background knowledge $\mathcal{K}$, and hence restricting according to $\mathcal{K}$ will change the adjacencies of $\mathcal{E}$ and force the equivalence class to encode $\mathcal{K}$. This is the issue also addressed in Remark 1. Even though Simple

TGES is optimal in the oracle setting, these issues arise when faced with statistical uncertainty from data. This issue does not occur in TGES as insured by Lemmas 3 to 5. By Figure 19 in Appendix A.4, we see that GES performs better in regards to the adjacency metrics than Simple TGES. Which backs up the hypothesis that the way Simple TGES restricts $\mathcal{E}$ with tiered background knowledge $\mathcal{K}$ is inadequate, as this restriction of $\mathcal{E}$ is the only difference between GES and Simple TGES.

TGES outperforms Simple TGES with respect to the metrics of directions as well, even though they are both given the same background knowledge and both use Meek's rules to infer extra orientation of edges. TGES has a higher concentration of high precision and high recall of directions. One wrongful prediction of an adjacency or direction might make Simple TGES or TGES infer even more wrongful directions by using Meek's rules. Hence this slightly worse performance of directing edges in Simple TGES might even stem from the issue of wrongful adjacencies.

When comparing TGES to the constraint-based alternative TPC in Figure 11, we see that TGES estimates the skeleton of the true tiered MPDAG more dependably than TPC. While the precision of adjacencies of TPC is on par with TGES, we see from the recall of adjacencies that TPC fails to predict as big a proportion of the true edges as TGES does. If we run TPC with a lower sparsity level, the predictions of TPC have a higher precision of adjacencies than TGES (see Figure 19 in Appendix A.4), but the recall of directions for the predictions of TPC simultaneously worsens even more as the sparsity level lowers.
However we see from the precision and recall of directions that TPC does a slightly better job at directing the edges compared to TGES. Keep in mind that the metrics of directions are strongly correlated with the predicted adjacencies as we are only looking at directions on edges that agree on adjacency with the true graph.

Overall, the comparison of TGES with the two alternative temporal causal discovery algorithms gives an indication of TGES working in practice and in certain aspects being the preferable choice to the constraint-based method TPC. Which method to use, depends on whether fitting the right skeleton or the right directions is of highest priority. As we will see in Section 5.2, given the information of tiers, nodes, density of the graph and sample size, the preferable choice of method will vary.

While a comparison of the directions of GES and TGES does not make much sense, since GES estimates CPDAGs and TGES estimates tiered MPDAGs, we can still look at how well the two different methods predict the skeleton of the true tiered MPDAG. The tiered background knowledge given no other information, does not restrict the skeleton of a graph it only restricts the directions given an adjacency. We do however see in Figure 12 that the tiered background knowledge, can improve the prediction of the skeleton of the graph. Both the precision and the recall of adjacencies improve by utilizing the tiered background knowledge in the way TGES does it. This result is not intuitive, and gives incentive to use tiered background knowledge in score-based methods for more than the trivial information stored in it, as it improves precision and recall of adjacencies. It is not obvious from current literature whether TPC and PC have the same relationship.

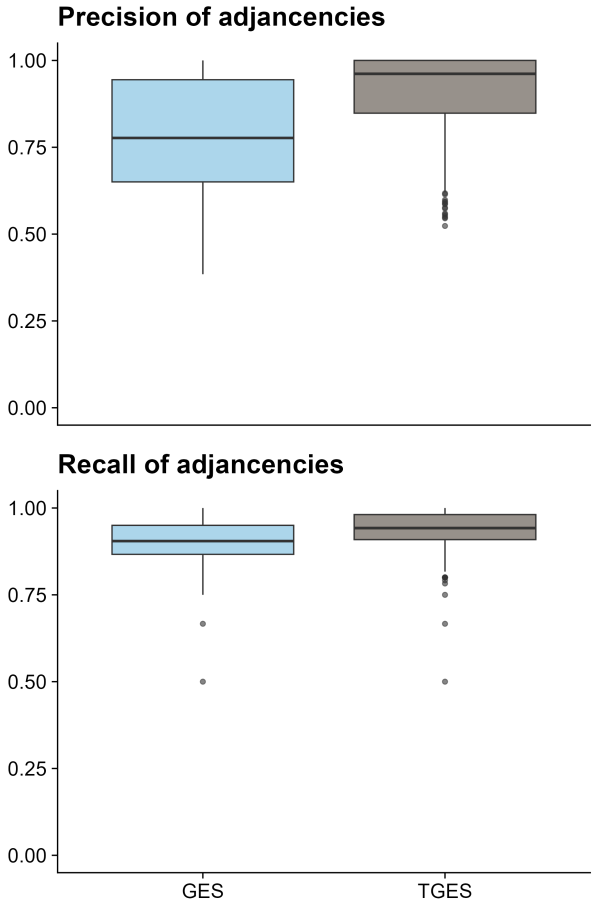


Figure 12: Precision and recall of adjacencies computed for 800 simulations. The methods included are TGES (black) and GES (blue)

5.1.2 In-tier directions

To assess the amount of extra directional information added to the predicted graph by having tiered background knowledge, we will compare the in-tier direction metrics of GES with TGES and Simple TGES. TPC with sparsity level 0.1 is included here to relate the constraint-based method's ability to the score-based method's ability to infer extra information from the tiered background knowledge. TPC has not been assessed according to in-tier directions before this thesis, and as we will discuss, seem to perform really well according to this new metric.

Even though we have simulated 800 data-generating DAGs, 48 of these contained at least one non-computable in-tier direction confusion matrix. An in-tier directional confusion matrix becomes non-computable due to no shared in-tier adjacencies[6] of the true tiered MPDAG and the estimated graph. If any one of the four methods could not compute the in-tier direction confusion matrix, the

[6]Might not even be any in-tier edges in the true data-generating DAG

entire simulation has been left out for better comparison of the methods. Hence Figure 13 is based on 752 simulations (see script ”Simulation study boxplots.Rmd” in Larsen, 2024)

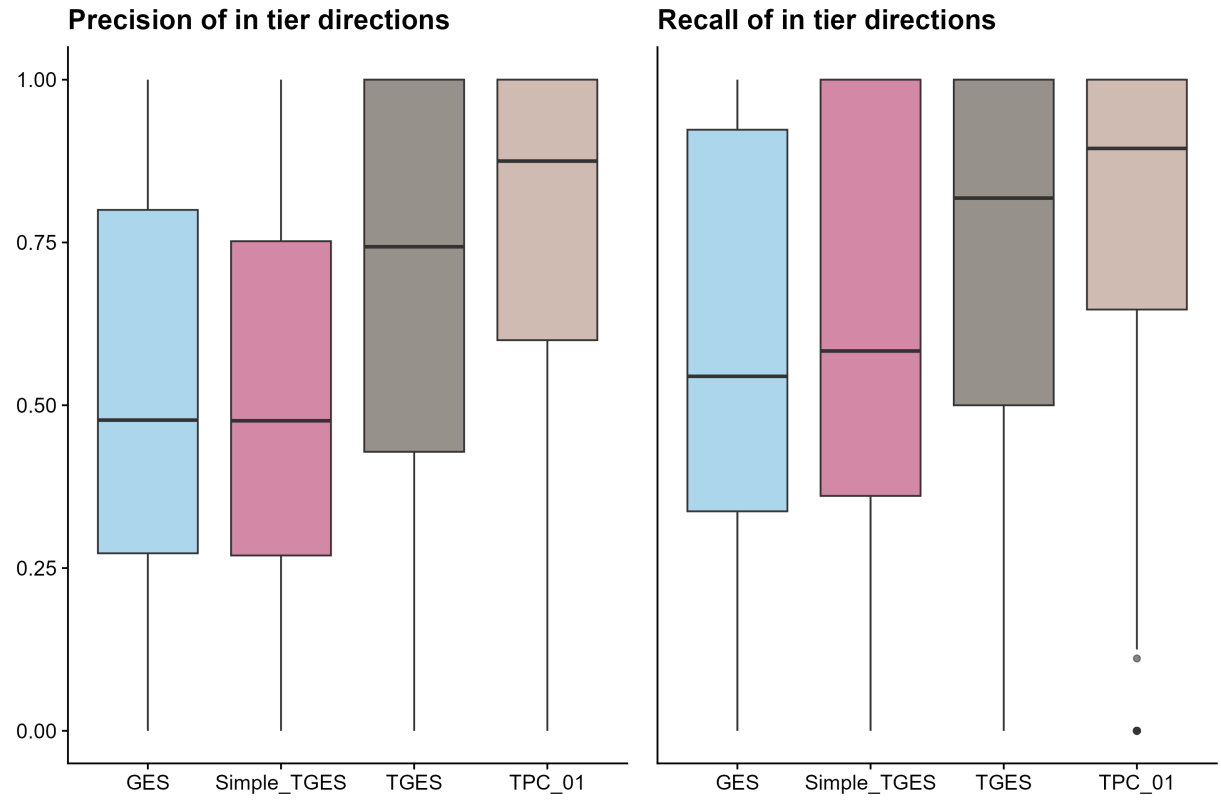


Figure 13: Precision and recall of in-tier directions for 752 simulations. The methods included are GES (blue), Simple TGES (red), TGES (black), and TPC with sparsity level 0.1 (light brown).

When comparing TGES to GES in terms of in-tier directional metrics (see Figure 13), we see that even though TGES is not given background knowledge on these edges, TGES is able to infer orientations that GES is not. This goes to show that adding tiered background knowledge gives us more than just the trivial edge orientations, as it helps orient the edges elsewhere in the graph. While this metric is made for comparing both predicted CPDAGs and predicted tiered MPDAGs to the true tiered MPDAG in a fairer way than the metric of all directions, the metric is still favoring in a way that it tends to score predicted tiered MPDAGs higher than CPDAGs. A predicted CPDAG that contains the true data-generating DAG might have a recall of in-tier directions less than 1, while the precision would always be 1.

TGES generally scores a higher precision and recall of in-tier directions than Simple TGES. We saw the same tendency when looking at the metric of all directions. We reach the conclusion that TGES is better at inferring the orientation of edges from the tiered background knowledge than

Simple TGES, both in general and inside the tiers.

When comparing GES and Simple TGES, we are actually evaluating how well the restriction of the equivalence class with the tiered background knowledge (Simple TGES) performs compared to the non restricted equivalence class (GES). In other words, we are evaluating the part of the algorithm outside the square in Figure 5. We see that the precision of in-tier directions worsens a bit when restricting the equivalence class, which most likely stems from the before addressed issues of the equivalence class not being guaranteed to be in agreement with the tiered background knowledge. The recall slightly betters when restricting the equivalence class. Overall, we do not see a clear pattern of inferring extra orientations of the edges when applying our tiered background knowledge in the way of Simple TGES.

TPC does a much better job at inferring the orientation of edges from the same tiered background knowledge. But as we saw in Figure 11, the recall of adjacencies is much lower for TPC, and the in-tier directions is a metric on the two nodes *given* a correctly predicted adjacency. Hence, TPC predicts fewer of the true adjacencies but manages to orient the edges it predicts more efficiently than TGES.

5.1.3 Conclusion

We see from our metrics that incorporating tiered background knowledge can be a valuable addition to a score-based method. We are able to infer more directions of edges than just the trivial directions given by the tiered background knowledge by utilizing the tiered background knowledge in the right manner. We see that not only are we able to better predict the orientations of the edges, but we also obtain some extra information on the skeleton of the true graph when incorporating tiered background knowledge into GES. This property would, in some settings, be of more value than the additional edges we are able to direct. However, this positive effect of incorporating tiered background knowledge into a score-based method is not the case for all implementations. We see that Simple TGES is outperformed by both TGES and TPC according to the chosen metrics. This motivates the use of TGES, instead of the simplest approach.
TGES proves to be a good alternative to TPC in the general setting, but the question on which of the two methods to use is not clear. We see that TPC struggles at predicting as high a proportion of the correct adjacencies as TGES (see top right in Figure 11) but it does a better job at directing edges correctly (see bottom half of Figures 11 and 13). This next section will dive into whether we

can choose one method over the other given information about the data-generating mechanism.

5.2 Performance of methods across parameters

The simulation study introduced in this section can be reproduced with the script ”Simulation study tables - parallel” (Larsen, 2024).
We will, throughout this simulation study, use the supplementary data from J. D. Ramsey and Andrews (2017) for comparison. This data is relevant since it includes evaluation of GES and PC using metrics similar to the ones used in this thesis.

In this second simulation study we wish to assess whether changing different factors of the data-generating mechanism affects how the methods perform. We have chosen to evaluate TGES, Simple TGES, GES, and TPC across 5 different parameters. In this section, TPC uses a Gaussian independence test and sparsity level 0.01. This simulation study structure allows for comparison within the method across parameters, and for comparison across methods. The parameters include sample size, sparsity of in-tier and cross-tier edges, number of nodes, and number of tiers. Each parameter can take two different values, which gives us 32 different combinations on which we evaluate our methods using the metrics introduced in Section 4.2.

$m \in (30, 1000)$. Sample size of dataset.

In-tier $\in (0.1, 0.9)$. Probability of inserting an in-tier edge in the underlying data-generating DAG.

Cross-tier $\in (0.1, 0.9)$. Probability of inserting a cross-tier edge in the underlying data-generating DAG.

#nodes $\in (7, 20)$. Number of nodes in the underlying data-generating DAG.

#tiers $\in (2, 5)$. Number of tiers in the underlying data-generating DAG.

We fit the methods on two different sample sizes m to evaluate the robustness of the methods on scarce and plentiful data. The small number of observations is set to 30 and the large number is set to 1000.
We also wish to assess data which comes from dense and sparse data-generating DAGs, furthermore we wish to differentiate between sparsity in cross-tier edges (sparsity parameter called ”Cross-tier”) and in-tier edges (sparsity parameter called ”In-tier”). The parameters ”Cross-tier” and ”In-tier” are equal to the possibility of there being an edge between two nodes in different tiers or in the same tier, respectively.

We have chosen to evaluate on two different number of nodes, denoted ”#nodes”. Either we assess our models on a dataset of 7 nodes, which is supposed to represent a ”small” true data-generating DAG, or on a dataset of 20 nodes that represents a ”big” true data-generating DAG. It could be argued that 20 nodes would be a small study as well, but the choice of the number of nodes to test also needs to be computationally feasible for the resources available. Other studies such as the one made by J. D. Ramsey and Andrews (2017) looks at no datasets smaller than 50 nodes, but all with relatively sparse data-generating DAGs compared to this simulation study.
The last parameter we evaluate across is the number of tiers, denoted ”#tiers”. #tiers differs between 2 and 5 tiers, and represent a finer and a coarser tiering of the nodes.

For each combination of parameters, we have simulated 100 different data-generating DAGs and 100 corresponding sets of data. On each simulation we compute the metric of interest and report the mean of the 100 simulations. For each metric of interest, we showcase the results in a table containing 32 means of the metric for each method.
We will go through the metrics of adjacencies, directions, and in-tier directions. We remind the reader that the difference from Section 5.1 is that we are now fixing the arguments we use for simulating the data and underlying data-generating DAG.

5.2.1 Metrics of adjacencies

We will begin by looking at the metrics of adjacencies, since the metrics of directions and in-tier directions depend on the prediction of true positive adjacencies, and therefore the prediction also depends on the prediction of adjacencies. We start off by showcasing precision of adjacencies in the table below.

Precision of adjacencies

			TGES				Simple TGES				GES				TPC			
		#nodes	7		20		7		20		7		20		7		20	
m	In-tier	Cross-tier\ #tiers	2	5	2	5	2	5	2	5	2	5	2	5	2	5	2	5
30	0.1	0.1	0.48	0.55	0.42	0.43	0.49	0.55	0.41	0.41	0.48	0.51	0.39	0.39	0.94	0.90	0.90	0.88
		0.9	0.81	0.96	0.66	0.77	0.72	0.85	0.49	0.52	0.77	0.89	0.57	0.68	0.83	0.99	0.70	0.70
	0.9	0.1	0.84	0.65	0.71	0.63	0.87	0.68	0.77	0.66	0.82	0.62	0.71	0.57	0.99	0.94	0.97	0.96
		0.9	0.94	0.97	0.88	0.91	0.94	0.93	0.87	0.85	0.93	0.93	0.87	0.87	0.99	0.99	0.97	0.98
1000	0.1	0.1	0.93	0.95	0.90	0.92	0.92	0.95	0.89	0.90	0.92	0.95	0.87	0.86	0.95	0.95	0.97	0.98
		0.9	0.99	1.00	0.92	0.97	0.84	0.78	0.47	0.56	0.87	0.85	0.57	0.71	0.94	0.99	0.72	0.77
	0.9	0.1	0.95	0.96	0.71	0.81	0.95	0.97	0.77	0.77	0.94	0.94	0.71	0.71	1.00	0.98	0.94	0.97
		0.9	0.93	0.98	0.89	0.90	0.92	0.92	0.88	0.87	0.92	0.92	0.88	0.88	0.98	0.99	0.93	0.96

Table 4: For each combination of parameters, the mean of the precision of adjacencies was evaluated on 100 simulations. Rows are divided among sample size (m), probability of an edge existing between two nodes in the same tier of the data-generating DAG (In-tier), and probability of an edge existing between two nodes in different tiers of the data-generating DAG (Cross-tier). Columns are divided among method used for prediction, number of nodes (#nodes), and number of tiers (#tiers) in the data-generating DAG. Lines and grey scale grading were added for a better overview.

Looking at Table 4, we notice that the relationship between the methods which we found in Figure 11 holds across most of the various combinations of parameters. With respect to precision of adjacencies, we see that GES and Simple TGES perform the worst out of the four and that TPC performs the best.

When looking at the supplementary data from J. D. Ramsey and Andrews (2017) on GES and PC, we see that PC, in general, has better precision than GES for more dense networks, and GES has relatively better precision on more sparse networks.
The same relation between the score-based and constraint-based method does not translate to the temporal evolution of the two methods. Instead, we observe that TPC has better precision across all combinations, except in the case of $m = 1000$, In-tier = 0.1, and Cross-tier = 0.9.

We notice that the score-based methods are sensitive to sample size on sparse networks. For $m = 30$ and lower In-tier and Cross-tier parameters, the precision of adjacencies is significantly lower compared to $m = 1000$. TPC manages to be quite robust across sample size relative to the score-based methods.

In Figure 12, we saw that TGES more precisely predicts the skeleton of the true graph than GES. We see in Table 4 that it is the case for every combination of parameters in this study that the precision of adjacencies of TGES is greater than or equal to the one of GES.

All methods worsen their precision of adjacencies as the number of nodes increases, while there is no clear relationship with the number of tiers in any of the methods.

Recall of adjacencies

			TGES				Simple TGES				GES				TPC			
		#nodes	7		20		7		20		7		20		7		20	
m	In-tier	Cross-tier\ #tiers	2	5	2	5	2	5	2	5	2	5	2	5	2	5	2	5
30	0.1	0.1	0.67	0.72	0.71	0.69	0.60	0.59	0.57	0.47	0.68	0.71	0.67	0.65	0.46	0.48	0.36	0.37
		0.9	0.56	0.64	0.36	0.41	0.43	0.28	0.21	0.14	0.52	0.46	0.29	0.28	0.20	0.32	0.05	0.07
	0.9	0.1	0.60	0.73	0.38	0.58	0.58	0.68	0.34	0.46	0.60	0.72	0.36	0.53	0.31	0.54	0.07	0.23
		0.9	0.48	0.61	0.30	0.34	0.38	0.30	0.22	0.16	0.44	0.45	0.26	0.26	0.16	0.32	0.04	0.06
1000	0.1	0.1	1.00	0.99	0.99	0.99	0.98	0.98	0.94	0.91	0.99	0.99	0.98	0.98	0.99	0.99	0.94	0.95
		0.9	0.98	0.98	0.90	0.92	0.84	0.44	0.49	0.32	0.93	0.80	0.75	0.71	0.76	0.88	0.24	0.27
	0.9	0.1	0.85	0.98	0.76	0.88	0.84	0.97	0.71	0.77	0.85	0.98	0.75	0.87	0.78	0.98	0.29	0.67
		0.9	0.79	0.93	0.72	0.76	0.60	0.45	0.52	0.37	0.75	0.75	0.71	0.70	0.55	0.81	0.18	0.25

Table 5: For each combination of parameters, the mean of the recall of adjacencies was evaluated on 100 simulations. Rows are divided among sample size (m), probability of an edge existing between two nodes in the same tier of the data-generating DAG (In-tier), and probability of an edge existing between two nodes in different tiers of the data-generating DAG (Cross-tier). Columns are divided among method used for prediction, number of nodes (#nodes), and number of tiers (#tiers) in the data-generating DAG. Lines and grey scale grading were added for a better overview.

In Figure 11, we saw that TGES predicts a bigger proportion of the true adjacencies than TPC, and when inspecting Table 5, we see that this is the case in every combination of parameters in this study. The difference between the two methods is bigger for denser graphs, where TPC tends to have a very low recall of adjacencies. In some cases the recall of adjacencies for TPC is as low as 0.04-0.07.

In the study performed by J. D. Ramsey and Andrews (2017) a similar relation is present between the score-based and the constraint-based method. We see that in J. D. Ramsey and Andrews (2017) the score-based method (GES) has slightly higher recall of adjacencies for sparse graphs, and much higher for denser graphs compared to the constraint-based method (PC).

All four methods worsen their recall of adjacencies as the number of nodes increases. The methods TGES and TPC perform better with more tiers, while the Simple TGES method generally worsens its recall of adjacencies when given more tiered background knowledge. It would be expected that the addition of tiered background knowledge has a positive effect on prediction power, but we see that for an inefficiently implemented algorithm such as Simple TGES, it can be harmful. We believe this behavior of the Simple TGES stems from the fact that GES is not guaranteed outside the oracle setting to find an equivalence class in agreement with the tiered background knowledge

(Remark 1).

Comparison of adjacencies:
Tables 4 and 5 exhibits a similar but more nuanced picture of the findings of the prediction powers of the skeleton we did from Section 5.1. First of all, even though the relative difference shifts across the parameters, the underlying question on which methods performs the best on each of the metrics of adjacencies, is consistent across the combinations of parameters included in this study.
The metrics on adjacencies make us think that TPC is less prone to add an edge than its score-based alternative[7]. This could explain the high precision, which then comes at the cost of a much smaller recall. TGES would therefore be a great alternative to TPC if the aim is to catch as many as the true underlying adjacencies as possible. In the case of a large data sample, TGES manages to do so without having an unreasonably low precision of adjacencies.
For the combinations of parameters where TPC has a significantly higher precision of adjacencies, it is generally also the case that the recall of adjacencies is quite low. There is no such thing as a free lunch, and in practice, we would choose which methods to use depending on the scope of our study.

Like in Figure 12, the recall and precision of adjacencies for TGES for all combinations of parameters (except one) is greater or equal than the one for GES. This gives us more confidence in choosing to add tiered background knowledge whenever we have the chance. Across all combinations, there are only positive benefits to including tiered background knowledge into a score-based approach. This is at least the case for an efficient implementation, and we see that if the tiered background knowledge is utilized differently, like Simple TGES, it can harm the prediction power of the skeleton of the true graph.

5.2.2 Metrics of directions

We look at metrics of directions of the tiered MPDAG in Tables 6 and 7. When we compare according to directions, we do not include wrongful or undetected adjacencies. Only edges where the true tiered MPDAG and the predicted graph agree on an adjacency affect the metric. As discussed earlier, the comparison of TGES with GES with respect to this measure is not meaningful, thus GES has been excluded from Tables 6 and 7, but will be included for metrics of in-tier directions in Tables 8 and 9.

[7]This property depends on sparsity level.

Precision of directions

			TGES				Simple TGES				TPC			
		#nodes	7		20		7		20		7		20	
m	In-tier	Cross-tier\ #tiers	2	5	2	5	2	5	2	5	2	5	2	5
30	0.1	0.1	0.82	0.98	0.74	0.93	0.77	0.95	0.66	0.88	1.00	1.00	0.97	0.99
		0.9	0.93	0.99	0.89	0.94	0.92	0.99	0.83	0.88	0.98	1.00	0.96	0.94
	0.9	0.1	0.53	0.74	0.42	0.70	0.40	0.65	0.38	0.57	0.99	0.99	0.91	0.87
		0.9	0.52	0.90	0.62	0.79	0.44	0.85	0.52	0.71	0.89	0.92	0.99	0.79
1000	0.1	0.1	0.97	0.99	0.96	0.99	0.96	0.99	0.91	0.97	0.99	1.00	0.99	0.99
		0.9	0.99	0.99	0.96	0.98	0.98	0.98	0.88	0.95	1.00	0.99	0.93	0.97
	0.9	0.1	0.46	0.99	0.45	0.74	0.42	0.93	0.40	0.60	0.64	0.99	0.54	0.80
		0.9	0.57	0.95	0.64	0.86	0.42	0.88	0.51	0.74	0.64	0.97	0.59	0.79

Table 6: For each combination of parameters, the mean of the precision of directions was evaluated on 100 simulations. Rows are divided among sample size (m), probability of an edge existing between two nodes in the same tier of the data-generating DAG (In-tier), and probability of an edge existing between two nodes in different tiers of the data-generating DAG (Cross-tier). Columns are divided among method used for prediction, number of nodes (#nodes), and number of tiers (#tiers) in the data-generating DAG. Lines and grey scale grading were added for a better overview.

First and foremost, we note that the combination of #nodes = 7 and #tiers = 5 in Table 6 contain mostly cross-tier edges, which by default gets directed by the tiered background knowledge, hence the precision of directions should be close to 1 by default.

We see from Table 6 that across all but a few combinations of parameters, TPC has a greater precision of directions than the score-based method. For a small sample size, TPC does a much greater job than the score-based alternative. TGES consistently has a better precision of directions than the Simple TGES method. The precision of TGES is sensitive to the in-tier density of the graphs, here meaning that a high in-tier parameter (0.9) results in a lower precision of directions. The Simple TGES method exhibits the same type of behaviour, while TPC only does for big sample sizes (m = 1000)[8].

[8]Precision of directions seem to get smaller for TPC for bigger sample size. This might be due to the metric relying on a correctly estimated adjacency.

Recall of directions

			TGES				Simple TGES				TPC			
		#nodes	7		20		7		20		7		20	
m	In-tier	Cross-tier\ #tiers	2	5	2	5	2	5	2	5	2	5	2	5
30	0.1	0.1	0.92	0.99	0.88	0.95	0.91	1.00	0.84	0.93	0.86	0.99	0.65	0.87
		0.9	0.98	1.00	0.92	0.96	0.98	1.00	0.87	0.92	0.94	1.00	0.29	0.76
	0.9	0.1	0.77	0.92	0.45	0.75	0.72	0.88	0.43	0.64	0.55	0.83	0.10	0.38
		0.9	0.70	0.96	0.63	0.81	0.65	0.92	0.57	0.77	0.45	0.92	0.03	0.36
1000	0.1	0.1	1.00	1.00	0.98	0.99	1.00	1.00	0.97	0.98	0.98	0.98	0.96	0.99
		0.9	1.00	1.00	0.97	0.99	0.99	0.99	0.92	0.97	0.99	1.00	0.92	0.98
	0.9	0.1	0.80	0.99	0.49	0.80	0.79	0.99	0.45	0.70	0.83	0.96	0.49	0.84
		0.9	0.74	0.98	0.69	0.88	0.63	0.95	0.58	0.80	0.79	0.99	0.56	0.82

Table 7: For each combination of parameters, the mean of the recall of directions was evaluated on 100 simulations. Rows are divided among sample size (m), probability of an edge existing between two nodes in the same tier of the data-generating DAG (In-tier), and probability of an edge existing between two nodes in different tiers of the data-generating DAG (Cross-tier). Columns are divided among method used for prediction, number of nodes (#nodes), and number of tiers (#tiers) in the data-generating DAG. Lines and grey scale grading were added for a better overview.

Once again, we note that the combination of #nodes = 7 and #tiers = 5 in Table 7 contain mostly cross-tier edges, which should result in the recall of directions being close to 1 by default.

In regards to Figure 11 in Section 5.1, we noted that, in general, Simple TGES has slightly lower metrics of directions than TGES, while TPC has slightly higher metrics of directions than TGES. The same relationship between Simple TGES and TGES is present across combinations of parameters here in Table 7, giving the indication that TGES does a better job at utilizing the tiered background knowledge to direct edges than its simpler alternative.

We do however discover that the ability of TPC to correctly direct edges falls dramatically when the sample size drops. In regards to recall of directions, TGES is relatively robust towards smaller sample sizes. We saw that for a sample size of $m = 30$, the precision of directions for TPC is much higher than that of TGES, but now we see that it comes at the cost of a recall of directions as low as 0.03-0.10.

Comparison of directions

From Tables 6 and 7 we see that while TPC have a prediction of adjacencies that is as good or slightly more accurate for larger sample sizes, almost none of the directed edges are being predicted correctly by the method when the sample size gets low.

In the case of large sample size the TPC and TGES seem equally good at directing edges. In the case of small sample size it makes sense to choose TGES as an alternative to TPC.

5.2.3 Metrics of in-tier directions

Next, we will introduce findings from the simulation study using the in-tier directions metrics. While we have computed the in-tier direction confusion matrix of all combinations of parameters, we have decided not to include the combination of #nodes = 7 and #tiers = 5 in Table 8 and Table 9. The true data-generating graph of this combination has between 2 and 3 possible in-tier edges, and the metric of in-tier directions, therefore, makes little sense to use on these graphs. We have chosen not to include Simple TGES, as the conclusions that could be made from comparing TGES and Simple TGES in the sense of in-tier directions are analogous to the ones made from all directions in Section 5.1. The excluded column and method can instead be found in an extended edition in Table 12 and Table 13 in Appendix A.4.

Precision of in-tier directions

			TGES			**GES**			**TPC**		
		#nodes	7	20		7	20		7	20	
m	In-tier	Cross-tier\ #tiers	2	2	5	2	2	5	2	2	5
30	0.1	0.1	0.67	0.48	0.65	0.52	0.43	0.55	0.98	0.96	0.94
		0.9	0.27	0.34	0.28	0.30	0.33	0.25	0.85	0.91	0.68
	0.9	0.1	0.46	0.37	0.52	0.38	0.33	0.43	0.92	0.95	0.71
		0.9	0.16	0.29	0.23	0.24	0.31	0.31	0.84	0.97	0.55
1000	0.1	0.1	0.94	0.89	0.92	0.88	0.78	0.86	1.00	0.95	0.95
		0.9	0.88	0.46	0.34	0.65	0.29	0.25	0.98	0.66	0.53
	0.9	0.1	0.41	0.38	0.58	0.36	0.35	0.44	0.57	0.42	0.67
		0.9	0.18	0.28	0.22	0.21	0.35	0.36	0.35	0.40	0.41

Table 8: For each combination of parameters, the mean of the precision of in-tier directions was evaluated on 100 simulations. Rows are divided among sample size (m), probability of an edge existing between two nodes in the same tier of the data-generating DAG (In-tier), and probability of an edge existing between two nodes in different tiers of the data-generating DAG (Cross-tier). Columns are divided among method used for prediction, number of nodes (#nodes), and number of tiers (#tiers) in the data-generating DAG. Lines and grey scale grading were added for a better overview. The combinations of #tiers = 5 and #nodes = 7 are omitted due to vague interpretation. The full table can be found in Appendix A.4

We see from Table 8 that as the true graphs get denser, the precision of in-tier directions gets smaller. Whether the graph gets denser with respect to cross-tier edges or in-tier edges yields the same result of a lower precision. This tendency is present across all of the methods included in this study.

We saw in Figure 13 that TPC infers in-tier edges better than TGES. We see from Table 8 that the same tendency is present across combinations of parameters, and TGES especially struggles in the case of small sample size, while TPC manages to keep a relatively high precision across all combinations.

We see from the relationship between GES and TGES that we, in most combinations of parameters, obtain extra correct in-tier edge orientations by utilizing tiered background knowledge. However, in the case of Cross-tier = In-tier = 0.9, GES has better precision than TGES, which suggests that there are settings where utilizing tiered background knowledge in the way of TGES harms the prediction power of inferring in-tier edge orientations. We could however argue whether it is even worth comparing such small values of precision to each other.

We see that whether TGES improves precision by knowing more tiers depends on the number of cross-tier edges. If fewer cross-tier edges, more tiers result in higher precision ($0.38 \rightarrow 0.58$ in Table 8) while for more cross-tier edges the relation inverses ($0.28 \rightarrow 0.22$ in Table 8). Whether the relation between number of cross-tier edges and change in precision of in-tier direction prediction with the increase of tiers is a monotonous relation or there is a tipping point, is not clear from this study, and further research would be needed.

Recall of in-tier directions

			TGES			GES			TPC		
		#nodes	7	20		7	20		7	20	
m	In-tier	Cross-tier\ #tiers	2	2	5	2	2	5	2	2	5
30	0.1	0.1	0.85	0.70	0.74	0.74	0.70	0.73	0.64	0.20	0.35
		0.9	0.83	0.42	0.39	0.78	0.43	0.37	0.73	0.09	0.16
	0.9	0.1	0.69	0.39	0.57	0.61	0.36	0.51	0.46	0.03	0.14
		0.9	0.46	0.32	0.27	0.36	0.34	0.35	0.42	0.01	0.10
1000	0.1	0.1	0.99	0.94	0.97	0.95	0.86	0.92	0.97	0.92	0.92
		0.9	0.95	0.51	0.43	0.90	0.39	0.42	1.00	0.67	0.65
	0.9	0.1	0.76	0.43	0.66	0.67	0.40	0.54	0.75	0.39	0.70
		0.9	0.41	0.32	0.26	0.42	0.41	0.43	0.62	0.38	0.46

Table 9: For each combination of parameters, the mean of the recall of in-tier directions was evaluated on 100 simulations. Rows are divided among sample size (m), probability of an edge existing between two nodes in the same tier of the data-generating DAG (In-tier), and probability of an edge existing between two nodes in different tiers of the data-generating DAG (Cross-tier). Columns are divided among method used for prediction, number of nodes (#nodes), and number of tiers (#tiers) in the data-generating DAG. Lines and grey scale grading were added for a better overview. The combinations of #tiers = 5 and #nodes = 7 are omitted due to vague interpretation. The full table can be found in Appendix A.4

Table 9 shows that the recall of in-tier directions for TGES worsens as edge density increases. Increased density of cross-tier edges and in-tier edges both worsen the recall.

While the precision of in-tier directions of TPC did much better than TGES for a small sample size, we see that this high precision also comes at the cost of a very low recall of in-tier directions. TPC predicts a high number of undirected in-tier edges than could be directed according to the true graph, this results in low recall but as we saw in Table 8 can still give a high precision.

GES has a higher recall of in-tier adjacencies than TGES for Cross-tier = In-tier = 0.9, while for most other combinations, TGES has a higher recall.

The most interesting observation from Tables 8 and 9 is that utilizing more tiered background knowledge, in some combinations of parameters, can actually lead to ”harmful” inferring of the direction of in-tier edges. We saw that for Cross-tier = 0.9, adding more tiers (going from 2 to 5)

will decrease both recall and precision of in-tier directions for TGES. If the case is also that In-tier = 0.9, we have that the best prediction of in-tier directions is obtained by not utilizing any tiered background knowledge (GES).
As we saw by looking at the metrics directions in Section 5.1.1, better orientation predictions are to be gained in general by having more tiers, but this information is primarily contained in the edges directly affected by the tiered background knowledge, and not the inferred in-tier directions.
The predictions of all methods tested in this study worsened as the true graph got denser. Especially interesting is the fact that there being more cross-tier edges has a negative effect on prediction of in-tier directions for the methods that take tiered background knowledge into account.
If we wish to use a score-based causal discovery algorithm and the focus is to correctly predict as large a proportion of in-tier directions as possible, this study would point towards choosing to utilize as little tiered background knowledge as possible. Meanwhile, this conclusion seems to go against theory, common sense, and the findings from Section 5.1. Hence, we argue that before we can make this conclusion, more research is needed into either 1. another definition of confusion metrics of directions, 2. the understanding of how more tiered background knowledge infers wrong directions, or 3. an alternative approach to implementing tiered background knowledge into a score-based causal discovery algorithm.

5.2.4 Conclusion

First of all, we have confirmed that the score-based method that utilizes tiered background knowledge tends to predict directions and adjacencies better than the score-based method which does not. We have seen that this holds despite of sample size and the structure of the data-generating DAG. However, we have also discovered that the ability to direct in-tier edges might weaken by utilizing tiered background knowledge.
Secondly, we have asserted that TGES is the most preferable temporal score-based causal discovery approach presented in this thesis. In any combination of parameters in this study we prefer TGES to its score-based alternative, Simple TGES, as our predictions only gain more correct information by utilizing the tiered background knowledge with TGES instead of Simple TGES. This is despite of the fact that Simple TGES is tiered optimal by Theorem 2.
Lastly, we have seen that the ability of TPC to correctly direct edges falls dramatically when the sample size drops, while TGES is relatively robust towards sample size in regards to the orientation of edges.

5.3 Computational time of TGES

This subsection will investigate computational efficiency of the TGES algorithm and compare it to the efficiency of GES and TPC. This subsection will compare overall computational time of the different methods in various ways. Even though TGES might outperform other methods in the before chosen metrics, computational time can, in some cases, be of equal importance as the precision and recall. While the main goal of the implementation has been to create a score-based causal discovery algorithm that gives meaningful estimations and not one that is computationally efficient, it is still relevant to address computational time and the possibilities for improving this. The scripts used for the following subsection are ”comp time of tges.Rmd” and ”Simulation study tables - parallel.Rmd” (Larsen, 2024).

5.3.1 Computational time over number of nodes

We will, throughout this subsection, compare different variants of the GES, TGES, and TPC methods. Note that Simple TGES is not included in the following subsection because its computational time is implied by the computational time of standard GES, plus one extra step of restricting to tiered background knowledge (see Figure 5), which has a relative computational cost equal to zero.

We have chosen to separate the time used by TGES into two and record the time of two versions of GES. Following is an overview of all the methods timed and discussed through this subsection:

TGES Step This is the collected time of the scoring step of the phases in TGES. This is where we go from a restricted equivalence class $\mathcal{E}^{\mathcal{K}}$ to a new equivalence $\mathcal{E}$, and corresponds to the GES step with the TBIC score criterion. In Figure 6, this is the time of going from tiered MPDAG to CPDAG.

TGES BG Knowledge This is the collected time of the restricting with background knowledge step of the phases in TGES. This is where we go from an equivalence class $\mathcal{E}$ to its restricted equivalence class $\mathcal{E}^{\mathcal{K}}$ by restricting with background knowledge $\mathcal{K}$ and using Meek's rule 1. This corresponds to recording the time of going from CPDAG through PDAG to Tiered MPDAG in Figure 6.

TGES All This is the sum of **TGES step** and **TGES BG Knowledge** and is the computational time of running the entirety of TGES to termination.

PCALG GES step This is the running time of GES from the pcalg package in R (Kalisch et al., 2024). This is called step since we do not have any restricting according to background knowledge in the phases.

GES Step This is analog to **TGES step** but it is run without given any background knowledge. This corresponds to running the GES, but implemented in exact same way as TGES. This is included to enhance the comparability of TGES and GES.

TPC This is the computational time of the TPC algorithm from R package tpc as implemented by (Witte, 2023), with sparsity level 0.01 and using a Gaussian independence test.

The computational time for each number of nodes is computed as the mean of 10 simulations with fixed #nodes. For each #nodes, #tiers are a uniformly drawn integer between 2 and $\min((\#\text{nodes}-1), 8)$, the weights of the edges are between 0 and 1, the probability parameters of in-tier and cross-tier edges are independently drawn from a uniform distribution $U(0, 1)$, the sample size is fixed to 1000. The nodes are assigned to the tiers in the following way: One node assigned to each tier and then remaining nodes randomly assigned to the tiers. All the data is drawn from a multivariate Gaussian distribution. The relevant methods are all given the same data for each simulation. The computational times are computed for $\#\text{nodes} \in \{3, 4, ..., 24, 25\}$ and are reported in seconds. See script "comp time of tges.Rmd" (Larsen, 2024) for code.

We start out by comparing **TGES All**, **TPC**, **GES step** and **PCALG GES step**. See Figure 14.

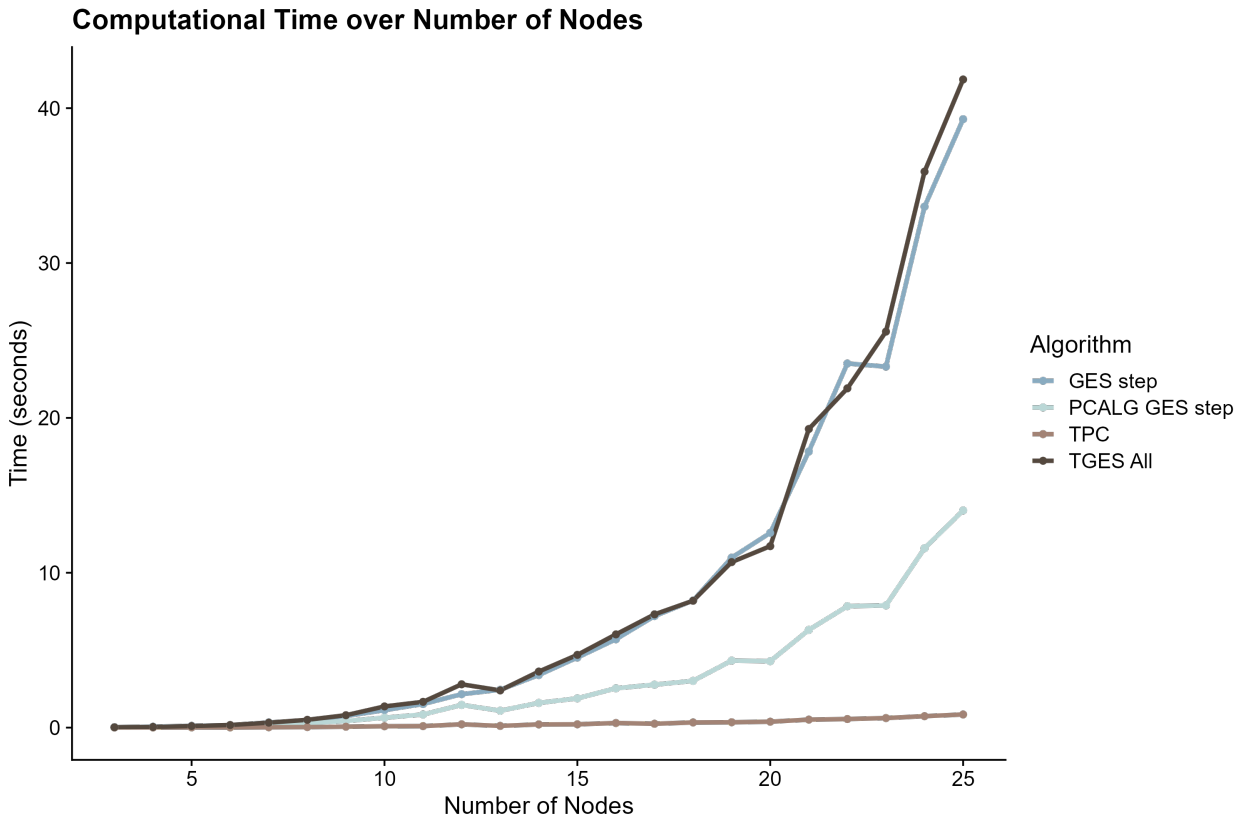


Figure 14: Computational time over number of nodes for **TGES All** (Larsen, 2024), **TPC** (Witte, 2023), **PCALG GES step** (GES by Kalisch et al., 2024) and **GES step** (GES by Larsen, 2024).

We see in Figure 14 that when it comes to computation time, the entirety of TGES (**TGES All**) performs poorly compared to **TPC** and **PCALG GES step**. The slower computation speed of **TGES All** and **GES step** compared to **PCALG GES step** might be due to a non-optimal implementation of the algorithm since the GES implemented by pcalg runs faster than **GES step** which is implemented in the same way as **TGES step**. **TPC** runs considerably faster than its score-based alternatives evaluated, but as can be seen in Figure 3 in Liu et al. (2024) the computation time of PC, which TPC is built on, has a lot to do with the conditional independence test, and can be made to run relatively much faster than GES (Liu et al., 2024) if choosing the right independence test. By examining the supplementary data from J. D. Ramsey and Andrews (2017) we see that GES and PC approximately have the same computation time, but in the case of dense graphs with high number of nodes, the computation time for GES is generally greater than that of PC. GES and PC in J. D. Ramsey and Andrews (2017) are using functions from the R package, pcalg (Kalisch et al., 2024).

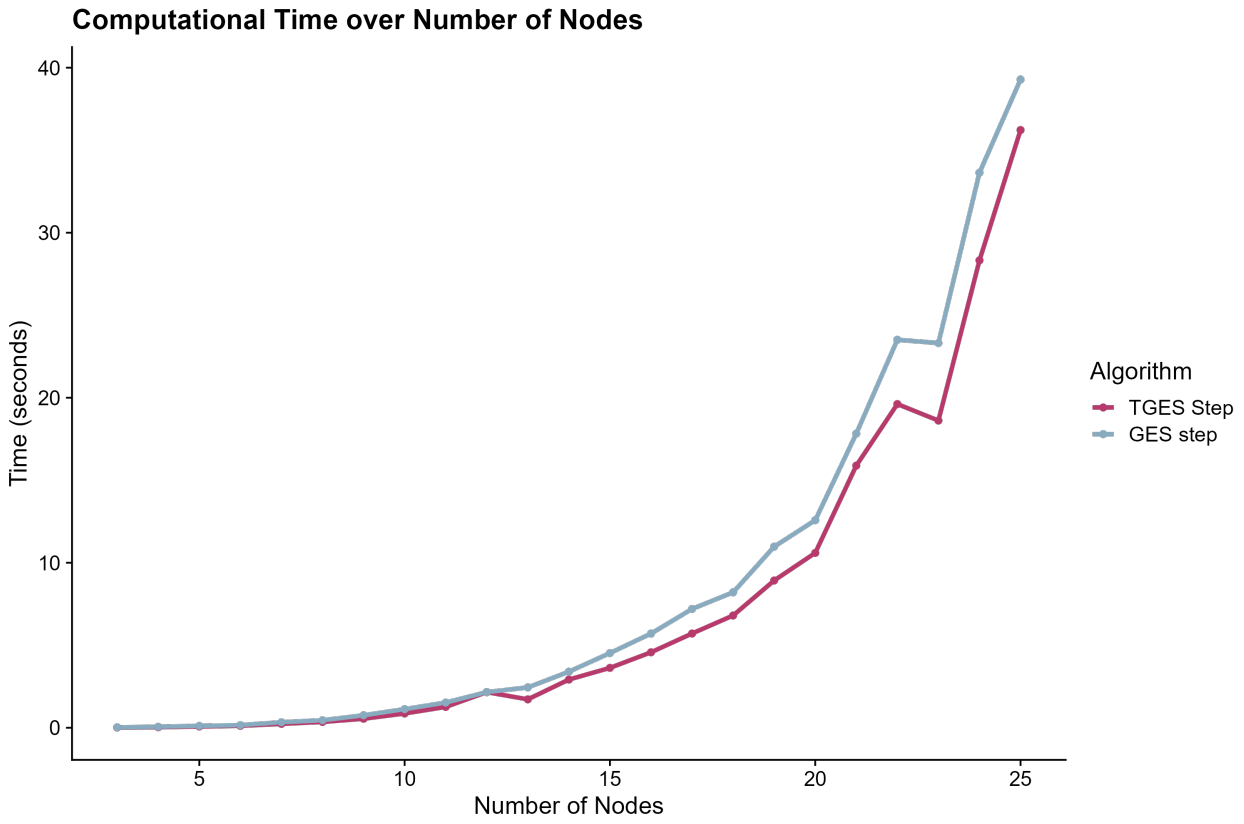


Figure 15: Computational time over number of nodes for **TGES All** (Larsen, 2024), **TGES step** (Step time of TGES by Larsen, 2024) and **GES step** (GES by Larsen, 2024).

To assess whether there is any possible computational gain of adding tiered background knowledge, we compare **GES step** and **TGES step** in Figure 15.

In this way, we will only be comparing equally good (or bad) implementations of TGES and GES and focus on whether adding background knowledge speeds up the forward, backward, and turning phases. As we can see from Figure 15, the steps of TGES are reliably faster than the steps of GES. This implies that adding tiered background knowledge decreases the time that the algorithm spends scoring graphs. Thus, if we can add tiered background knowledge without any loss of computational efficiency, TGES has the potential to run faster than GES. For this to make sense, the forward, backward, and turning phases of TGES would need to be implemented in a faster way as well, since this part is responsible for the majority of the computational time in the current implementation.

Judging by these plots, TGES runs slower than all of its alternatives, which seem to root from the way TGES has been implemented and the computational cost of restricting according to tiered background knowledge. Assuming that the computational cost of restricting according to tiered

background knowledge can be implemented more efficiently and that we can implement TGES in a faster way, TGES has potential to run faster than standard GES.

5.3.2 Across parameters

While TGES is slower than GES when averaging over different sample sizes and structures of the underlying DAG, we also wish to assess the computational efficiency across different combinations of parameters.

The parameters used in this subsection are equal to the ones used in Section 5.2, and the script for reproducing the results is ”Simulation study tables - parallel.Rmd” (Larsen, 2024).

In this section, we will not differentiate between time spent in different parts of TGES, nor will we run GES with other implementations than the one by pcalg.

Computational time

			TGES All				**PCALG GES step**				**TPC**			
		#nodes	7		20		7		20		7		20	
m	In-tier	Cross-tier\ #tiers	2	5	2	5	2	5	2	5	2	5	2	5
30	0.1	0.1	0.34	0.30	19.22	20.57	0.23	0.23	5.63	6.01	0.02	0.02	0.05	0.05
		0.9	0.80	1.04	27.58	46.56	0.49	0.55	10.04	18.38	0.03	0.04	0.10	0.27
	0.9	0.1	0.91	0.61	35.54	33.59	0.66	0.49	17.97	13.85	0.05	0.04	0.18	0.13
		0.9	1.08	1.27	36.20	38.92	0.79	0.82	18.32	18.38	0.05	0.06	0.21	0.29
1000	0.1	0.1	0.46	0.39	19.95	18.21	0.37	0.34	6.80	7.37	0.04	0.04	0.25	0.18
		0.9	0.96	1.90	3.39	1.41	0.60	1.11	22.13	26.24	0.10	0.13	3.54	3.74
	0.9	0.1	1.20	0.60	37.75	39.85	0.98	0.49	23.99	18.09	0.11	0.05	1.56	0.96
		0.9	1.74	1.97	3.98	2.32	1.05	1.13	28.73	29.95	0.24	0.16	2.00	2.28

Table 10: For each combination of parameters, the mean of the computational time of 100 simulations is reported in the table. Rows are divided among sample size (m), probability of an edge existing between two nodes in the same tier of the data-generating DAG (In-tier), and probability of an edge existing between two nodes in different tiers of the data-generating DAG (Cross-tier). Columns are divided among method used for prediction, number of nodes (#nodes), and number of tiers (#tiers) in the data-generating DAG. Lines and grey scale grading were added for better overview.

The computational time of TPC is, in almost every instance, much lower than the time for the score-based methods. The difference between TGES and TPC is consistently huge for sample size $m = 30$ and for some combinations[9] of parameters, the time of running TGES is as much as 380 times as big.

[9]#tiers = 2, #nodes = 20, In-tier = Cross-tier = 0.1 and $m = 30$

For bigger sample size $m = 1000$, the time of TGES depends hugely on the number of cross-tier edges and tiers. The difference is clearest when looking at graphs with 20 nodes. More tiered background knowledge or more edges affected by tiered background knowledge makes TGES run faster. With a small amount of tiered background knowledge and a small amount of cross-tier edges, TGES runs much slower than the other algorithms.
For the instance of #tiers = 4, #nodes = 20, In-tier = 0.1, Cross-tier = 0.9, and $m = 1000$, we observe a faster run time for TGES than TPC. But if we change the cross-tier to 0.1 instead, TGES will be 10 times slower, while TPC will get 20 times faster.

The computation time of TPC and GES are sensitive to #nodes, sample size, and density of edges in the graph, but we see that for TGES, the amount of tiered background knowledge and the number of edges to use the tiered background knowledge on plays a crucial part in the computational time. Only when we are in the scenario of large sample size and large amounts of tiered background knowledge and cross-tier edges, does TGES have a chance at running faster than TPC.

TGES only runs quicker than GES in these ideal scenarios, where there is a lot of tiered background knowledge, lots of cross-tier edges, large sample size, and lots of nodes. If any of these criteria are not fulfilled, TGES runs much slower than GES.
This does however show the potential of big savings on computation time when we know we are dealing with a complex networks with many nodes and many edges.

6 Discussion

This section will discuss the findings and methods used throughout this thesis. We will start by comparing different causal discovery algorithms introduced, then we will further discuss limitations and alternatives to TGES. Next, we will discuss the limitations that could be argued of for the simulation study. Lastly, we present open questions put forth by this thesis and possible ideas on how to tackle them.

6.1 Comparison of causal discovery algorithms

We were hoping to be able to create an algorithm that by using tiered background knowledge would be able to create a better prediction than if we were not to use background knowledge. We saw that in general TGES has better adjacency and in-tier direction performance metrics than GES (Figures 12 and 13). This was once again confirmed when we checked across different parameters, and saw that the tendency of TGES beating GES was consistent across combinations of parameters. We saw that in the case of the example presented in Section 3.4.3, GES managed to wrongly direct some edges but overall did a good job estimating the skeleton. This example might be more challenging to estimate for GES since 6/7 edges are cross-tier edges, and there are no v-structures in the underlying data generating DAG.
While the computation time of TGES is a bit longer, this is mostly due to the time it takes to restrict according to $\mathcal{K}$. The comparison of the time of the steps in GES and TGES (Figure 15) showed promise since the search space of neighboring equivalence classes was shrunk for TGES.
We conclude that we always recommend utilizing tiered background knowledge whenever we are able to, because it seems to give better predictions across different scenarios. This is of course with the assumption of the tiered background knowledge we use being true.

Even with true tiered background knowledge we are not guaranteed a good performance across methods that use this tiered background knowledge. We saw that even if the method is optimal, the performance can be bad outside of the oracle scenario. We saw in Theorem 2 that in the limit of large sample size the method Simple TGES would correctly estimate the true tiered MPDAG, which in turn contained more than or equal the amount of directional information as the best possible estimate by GES. It is no surprise that by using more true information we can make a better estimate, while it is less intuitive that this was not always the case outside the limit of large sample size. In the simulation study, Simple TGES was, in many aspects, outperformed by GES since

the restricting by true tiered background knowledge on a graph that is not in agreement with the true tiered background knowledge can do more harm than doing nothing. We, for instance, saw an example in Section 3.4.3 where the prediction by GES had more correct adjacencies than the prediction by Simple TGES.

Simple TGES works in the oracle scenario, but outside of that, we cannot be sure, and we see that the connection between being optimal and working in practice is not that strong. For example, we could make a variant of Simple TGES, called *Forbidden TGES* that would when the equivalence class returned by GES is not in agreement with $\mathcal{K}$, the prediction is the graph of forbidden edges. If the equivalence class is in agreement with $\mathcal{K}$, we restrict according to $\mathcal{K}$ and infer edges with Meek's rule 1. The method Forbidden TGES could be proved to be optimal in the limit of large sample size in the same as Theorem 2, but would yield terrible results when measuring its performance outside of the oracle scenario.

This observation tells us that the action of removing edges contradicting $\mathcal{K}$ could be made to be a lot worse while still keeping the result of optimality (Theorem 2). This could also imply that we could improve this part of Simple TGES and still keep optimality of the method. Meanwhile, with the current implementation, we can conclude that TGES is the better alternative when wanting to do score-based causal discovery with tiered background knowledge.

We saw in Figure 4 in Section 3.4.3 that TPC and TGES both reach the same prediction of the true tiered MPDAG. It has been proven in Bang et al. (2024) that TPC is sound and complete in the oracle setting, while it remains a conjecture for TGES in this thesis. We saw from the simulation study that TPC with a Gaussian independence test could run much quicker than TGES in general, but also that in the setting of large sample size, large amounts of tiered background knowledge, and cross-tier edges, TGES has the potential to be faster than TPC (Table 10).

The TPC predicts directions and in-tier directions with more precision than TGES but manages to predict a smaller proportion of the true directions (Tables 6 and 7). We saw in Table 7 that in one combination of parameters, TPC only managed to correctly predict 3% of the correct directions. In contrast, TGES in the same setting predicted 63% of the correct directions. TPC and TGES predict adjacencies with similar precision (Table 4), but TPC picks up on a much lower proportion of the true adjacencies that TGES (Table 5 and fig. 11). These observations seem to be consistent across the sparsity level of TPC.

The sparsity parameter of TPC allows for tuning the density of the edges added to the prediction, hence if we have an idea on the true density we can calibrate with the sparsity parameter. The implementation of TGES in (Larsen, 2024) does not allow for any tweaking of the sorts. Whether it is beneficial to have the opportunity to tune your prediction sparsity depends on the study design and the researcher's preferences.

6.2 Discussion of Temporal score-based algorithms

The final implementation of TGES presented in this thesis is derived from a focus of practical use. The Lemmas 3 to 5 are results that ensure that we stay within classes that are in agreement with $\mathcal{K}$, but we were not able to prove optimality of TGES, even though we expect it to be true (Conjectures 1 and 2). As for now, we are not able to show that there is a path of greedy steps using the TBIC from a $\mathcal{G}$ that contains p to a $\mathcal{G}^*$ that is a perfect map of p. We have by Theorem 1 that a path exists, but we are not guaranteed that such a path only has steps that encode $\mathcal{K}$. Hence, we would need an extension of Theorem 1 along the lines of the following conjecture to be able to prove Conjecture 2 in an analogous way of the proof of Lemma 2.

Conjecture 3. *(**Tiered version** of Theorem 1) Let $\mathcal{G}$ and $\mathcal{H}$ be any pair of DAGs such that $\mathcal{G} \leq \mathcal{H}$ **and they both encode tiered background knowledge** $\mathcal{K}$. Then, there exists a sequence of edge reversals and additions in $\mathcal{G}$ with the following properties:*

1. *Each edge reversed is a covered **in-tier** edge*
2. *After each reversal and addition $\mathcal{G}$ is a DAG and $\mathcal{G} \leq \mathcal{H}$*
3. *After all reversals and additions $\mathcal{G} = \mathcal{H}$*

where $\mathcal{G} \leq \mathcal{H}$ refer to that every independence relationship in $\mathcal{H}$ holds in $\mathcal{G}$ and a covered edge is an edge that in its Markov equivalence class cannot be directed.

To prove this conjecture, we would need to do research in graph theory in a way that had not been the aim of this thesis. While Conjecture 3 would make us able to prove Conjecture 2, how we would prove Conjecture 1 is less clear.

As seen, TGES assumes score equivalence. This assumption is true when working with linear Gaussian data. It has been shown that a score criterion that takes non-linearity into account may not score all DAGs in the same equivalence class the same, i.e., the scoring criterion might not be score equivalent (Huang et al., 2018). Hence it would be the same case for a score criterion that takes tiered background knowledge and non-linearity into account. GES, as implemented by pcalg (Kalisch et al., 2024), might not even be able to find an equivalence class due to the risk of running in a loop like in Remark 2. If the local score of $X \rightarrow Y$ is different from $Y \rightarrow X$, and they both belong to the same equivalence class, we might keep adding and removing the same edge forever.

We chose not to develop a method that would restrict according to $\mathcal{K}$ between phases. This would

bring up issues no matter which of the scoring criteria presented in the thesis we were to use. If we were to use BIC as a score criterion, we would run into an issue similar to the one addressed in Remark 1, and we could not be guaranteed that we would be able to restrict the equivalence class without changing the skeleton. If we instead used the TBIC score criterion, we could run into the issue addressed in Remark 2 of adding and removing the same undirected cross-tier edge forever. Hence, we deemed it most efficient to develop an algorithm that would use the tiered background knowledge every step.

When attempting to deal with the issue described in Remark 2 one other alternative was tried out, *GES with complete cross-tier edges*. The idea was that instead of initializing with an empty graph, we would initialize with a graph with all cross-tier directed edges allowed by the tiered background knowledge. Then, the forward phase would be run only on in-tier edges by locally scoring with TBIC, while the backward and turning phases would be run on the whole graph by locally scoring with TBIC. In this way, once we removed a cross-tier edge, the algorithm would never be able to add it again. The algorithm turned out to give some reasonable results at times and also avoided the issue of Remark 2, but not being able to add cross-tier edges and starting out with a very dense graph made the performance of the method unreliable.
The pseudo code for the algorithm is included in Appendix A.5 in Algorithm 5.

6.2.1 Implementation of TGES

As mentioned earlier, the focus of the implementation of TGES has been to create an algorithm that predicts well, and there has been less focus on computational efficiency and whether it is easy to use. Going forward, if the ”tges()” function (Larsen, 2024) should be user-friendly some structural changes to make it more straightforward and faster should be made. For example, the current implementation is checking for Meek’s rules 1-4 (Meek, 1995), but we have from Lemmas 3 to 5 that it would be enough to use Meek’s rule 1. For extra speed we could rewrite the code in C++, as have been done with ”ges()” from pcalg (Kalisch et al., 2024). One last desirable extension to ”tges()” is the ability to take in data that is either binary or nominal.

6.3 Simulation study limitations

While the simulation study was sufficient for the aim of this thesis, multiple critique points could be made. We are, when choosing an interval for the uniform distribution from which we draw our parameters, also making a choice not to include any graphs with parameters outside the interval. For example, we are not evaluating any graphs with more than 20 nodes in Section 5.1, thus inad-

vertently excluding all graphs of larger size. We despite of this believe the parameter space to be of sufficient size for a preliminary study into the performance of these new methods.
In the simulation study in Section 5.2 we utilized the possibility of distinguishing between density of in-tier and cross-tier edges. While this gave us a simulation study that was more sensitive to the two different kinds of density of edges, it also brought with it some issues when trying to infer the effect of number of tiers on the metrics. For example, with a combination of low cross-tier density and high in-tier density, the number of tiers and nodes played a huge role on the overall density of the graph by changing the set of possible cross-tier and in-tier edges. Thus, for this setting, it is not possible to isolate whether the change in metrics is caused by the change in number of tiers or the change in the overall density of the graph.
Another possible limitation of the simulation study is the choice of metrics. We have that recall and precision of directions and in-tier directions are not rewarding the ability to correctly predict undirected edges, while it does however punish for not being able to do so (See Tables 2 and 3). If we would like to account for the number of TN in some way we could measure the algorithm's ability to do this as well. In the same way, recall and precision of adjacencies is not able to pick up on correctly estimated non-adjacencies (See Table 11). Hence, we know very little of how well the methods predict non-adjacencies, which could be a problem in situations where the information about where there are no causal relationships is of higher value than the information on where there are causal relationships.

In the simulation study, all data were drawn from multivariate Gaussian distributions, as the focus was testing the performance of TGES on the setting were everything is ideal except the sample size. Meanwhile, data from the real world does not look like this and another simulation study into the performance of TGES on non-Gaussian data would be of great value for discovering when TGES is a valid option to use. Such a simulation study was not the focus of this thesis but could be in future research.

6.4 Future research questions

In regards to future research, we first of all propose further work into making TGES faster, such that it can become a viable option for estimating graphs with many variables. Earlier work has looked into creating faster ways of computing GES (M. Chickering, 2020; J. Ramsey et al., 2017). A similar implementation of TGES would expand the use cases of TGES. Another approach to making TGES faster is to minimize the number of edges which are scored. Instead of creating a score like TBIC that penalizes wrongly directed edges, we could completely ignore scoring wrongly directed edges. In the current implementation of TGES, we are searching over a reduced search space in

the backward phase but not in the turning and forward phases.
Another possibility of increasing the speed of TGES is utilizing graph theory to avoid having to estimate a CPDAG each step which is then restricted according to $\mathcal{K}$ and extra edges directed by Meek's rule 1. If we could have a step that went from a tiered MPDAG straight to another tiered MPDAG, it would most likely increase the computation speed as well.

In this thesis we did not look into any applications of TGES on real world data, such as life-course studies. In many cases data from life-course studies would be a mix between categorical and continuous data, hence a further study into how TGES could deal with this would need to be made prior.

As TPC has a sparsity parameter, we would like to be able to have a similar way of tuning GES. A possible way of doing so could be to scale the penalty term of BIC (Equation (2)) by adding on a factor λ such that $S_B(\mathcal{G}, \mathbf{D}) = \log p(\mathbf{D}|\hat{\theta}, \mathcal{G}) - \lambda \frac{\#\text{parameters}}{2} \log m$. What effect this would have on the performance of estimation would need to be further studied.
We also propose to look into whether the method of TGES could be extended to take in other forms of background knowledge. We have in this thesis only studied tiered background knowledge, but some of the new developments, like TBIC, could easily be extended into taking any kind of background knowledge into account. Tiered background knowledge does have some needful properties (Bang & Didelez, 2023) like not introducing cycles, but this would mean we had to utilize all 4 of Meek's rules.

The last research question we propose is the question of the use of tiered background knowledge which we cannot be certain is true. Through this thesis, we have assumed all tiered background knowledge to be true, and TGES ranks the information given by $\mathcal{K}$ higher than the information induced by the data. If we could not be certain that the tiered background knowledge was true, we might have to check it with the data first or only be able to use it as an initial guess of GES. Another idea was to penalize graphs contradicting $\mathcal{K}$ as something less serious than $-\infty$. The question of whether tiered background knowledge would even be of value if it was not always true, would also need to be further studied.

The aim of this thesis was to develop a score-based algorithm that could incorporate tiered background knowledge and then assess whether it would be a viable alternative to the TPC algorithm by Petersen et al. (2021). We want to finish up by stating that the method TGES and the implementation "tges()" in R (Larsen, 2024) is a sensible alternative to TPC and shows the potential to not only have reasonable prediction performance but also could become computationally efficient.

References

Bang, C. W., & Didelez, V. (2023). Do we become wiser with time? on causal equivalence with tiered background knowledge.

Bang, C. W., Witte, J., Foraita, R., & Didelez, V. (2024). *Improving the finite sample performance of causal discovery by exploiting temporal structure* [Unpublished].

Chickering, D. M. (2003). Optimal structure identification with greedy search. *J. Mach. Learn. Res.*, *3*(null), 507–554. https://doi.org/10.1162/153244303321897717

Chickering, M. (2020, March). Statistically efficient greedy equivalence search. In J. Peters & D. Sontag (Eds.), *Proceedings of the 36th conference on uncertainty in artificial intelligence (uai)* (pp. 241–249, Vol. 124). PMLR. https://proceedings.mlr.press/v124/chickering20a.html

Hasan, U., & Gani, M. O. (2023). Kgs: Causal discovery using knowledge-guided greedy equivalence search.

Haughton, D. M. A. (1988). On the choice of a model to fit data from an exponential family. *The Annals of Statistics*, *16*(1), 342–355. Retrieved March 4, 2024, from http://www.jstor.org/stable/2241441

Hauser, A., & Bühlmann, P. (2012). Characterization and greedy learning of interventional markov equivalence classes of directed acyclic graphs.

Hernan, M., & Robins, J. (2020). *Causal inference: What if*. CRC Press. https://books.google.dk/books?id=_KnHIAAACAAJ

Huang, B., Zhang, K., Lin, Y., Schölkopf, B., & Glymour, C. (2018). Generalized score functions for causal discovery. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1551–1560. https://doi.org/10.1145/3219819.3220104

Kalisch, M., Hauser, A., Maathuis, M. H., & Mächler, M. (2024). *An overview of the pcalg package for r* [R package version 4.3.2]. https://cran.r-project.org/package=pcalg

Kummerfeld, E., Williams, L., & Ma, S. (2024). Power analysis for causal discovery. *International Journal of Data Science and Analytics*, *17*(3), 289–304. https://doi.org/10.1007/s41060-023-00399-4

Larsen, T. E. (2024, April). *TGES* (Version 1.0.0). https://github.com/tobiaselar/TGES

Liu, W., Huang, B., Gao, E., Ke, Q., Bondell, H., & Gong, M. (2024, January). Causal discovery with mixed linear and nonlinear additive noise models: A scalable approach. In F. Locatello & V. Didelez (Eds.), *Proceedings of the third conference on causal learning and reasoning* (pp. 1237–1263, Vol. 236). PMLR. https://proceedings.mlr.press/v236/liu24b.html

Meek, C. (1995). Causal inference and causal explanation with background knowledge. *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, 403–410.

OEIS Foundation Inc. (2023). A003024: Number of acyclic digraphs (or dags) with n labeled nodes [Accessed: 13-05-2024].

Pearl, J. (1988). Chapter 3 - markov and bayesian networks: Two graphical representations of probabilistic knowledge. In J. Pearl (Ed.), *Probabilistic reasoning in intelligent systems* (pp. 77–141). Morgan Kaufmann. https://doi.org/https://doi.org/10.1016/B978-0-08-051489-5.50009-6

Peters, J., Janzing, D., & Schlkopf, B. (2017). *Elements of causal inference: Foundations and learning algorithms*. The MIT Press.

Petersen, A. H., Osler, M., & Ekstrøm, C. (2021). Data-driven model building for life course epidemiology. *American Journal of Epidemiology*, *190*. https://doi.org/10.1093/aje/kwab087

Petersen, A. H. (2022). *Causaldisco: Tools for causal discovery on observational data* [R package version 0.9.1]. https://CRAN.R-project.org/package=causalDisco

Raghu, V. K., Poon, A., & Benos, P. V. (2018). Evaluation of causal structure learning methods on mixed data types. *Proceedings of machine learning research*, *92*, 48–65. https://api.semanticscholar.org/CorpusID:53598779

Ramsey, J., Glymour, M., Sanchez-Romero, R., & Glymour, C. (2017). A million variables and more: The fast greedy equivalence search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images. *International Journal of Data Science and Analytics*, *3*(2), 121–129. https://doi.org/10.1007/s41060-016-0032-z

Ramsey, J. D., & Andrews, B. (2017). A comparison of public causal search packages on linear, gaussian data with no latent variables. *CoRR*, *abs/1709.04240*. http://arxiv.org/abs/1709.04240

Schwarz, G. (1978). Estimating the Dimension of a Model. *The Annals of Statistics*, *6*(2), 461–464. https://doi.org/10.1214/aos/1176344136

Spirtes, P., Glymour, C., & Scheines, R. (1993, January). *Causation, prediction, and search* (Vol. 81). https://doi.org/10.1007/978-1-4612-2748-9

Witte, J. (2023). *Tpc: Tiered pc algorithm* [R package version 1.0]. https://CRAN.R-project.org/package=tpc

Zheng, Y., Huang, B., Chen, W., Ramsey, J., Gong, M., Cai, R., Shimizu, S., Spirtes, P., & Zhang, K. (2024). Causal-learn: Causal discovery in python. *Journal of Machine Learning Research*, *25*(60), 1–8.

A Appendix

A.1 Computational time graph

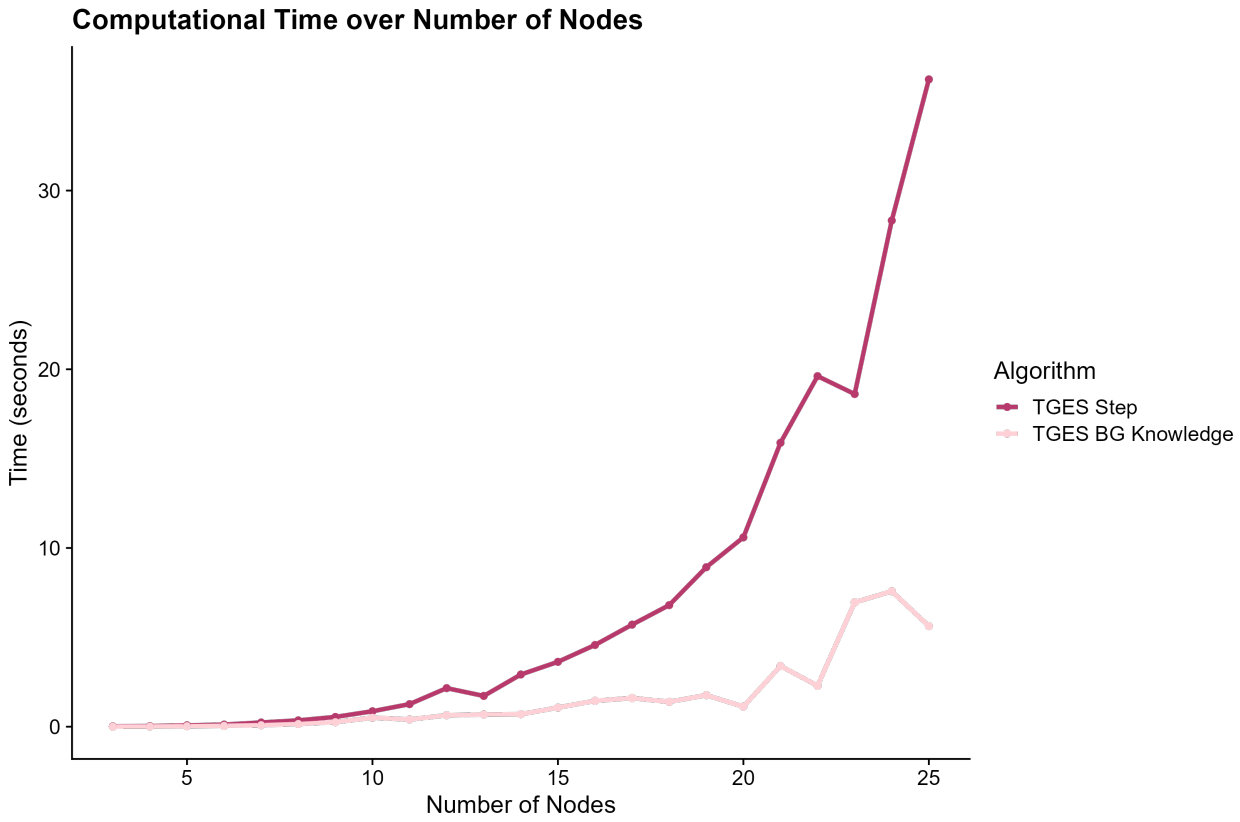


Figure 16: Computational time over number of nodes for TGES step (Step time of TGES by Larsen, 2024) and TGES BG Knowledge (Larsen, 2024) (Time used restricting according to $\mathcal{K}$ and infering edges using Meek's rule 1).

A.2 Meek's rules

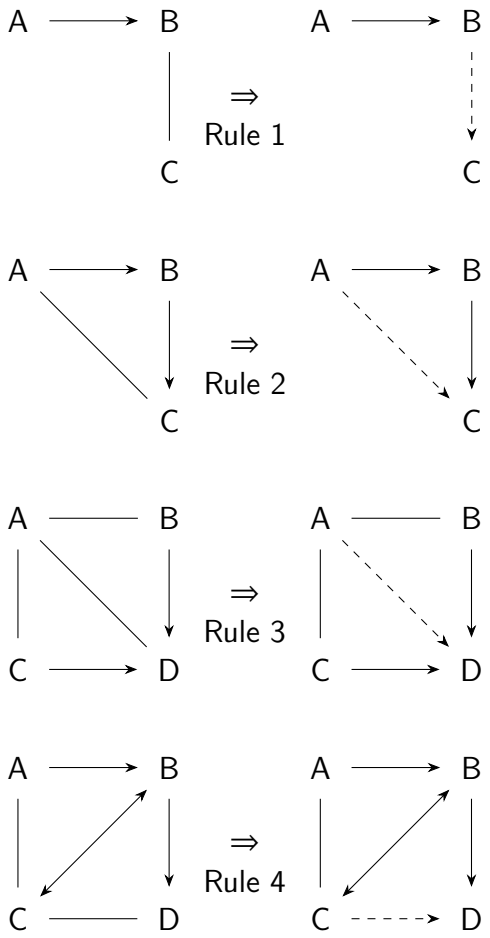
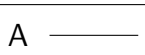
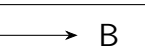
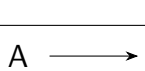
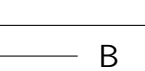
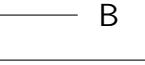
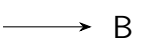
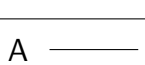
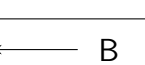
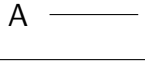
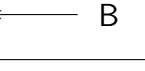
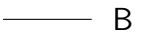
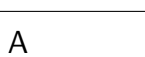
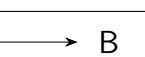
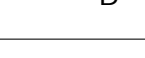


Figure 17: Representations of Meek's rules 1-4 (Meek, 1995). Dashed edge is infered by the rule.

Note here that an undirected edge is marked as — and an edge that can be undirected or directed either way is marked as $\longleftrightarrow$.

A.3 Confusion matrix according to adjacencies

Table 11: Overview of values in confusion matrix for adjacencies

Estimated edge	True edge	Result
		TP
		TP
		TP
		TP
		TP
		FN
		FN
		TN
		FP
		FP

A.4 Simulation study

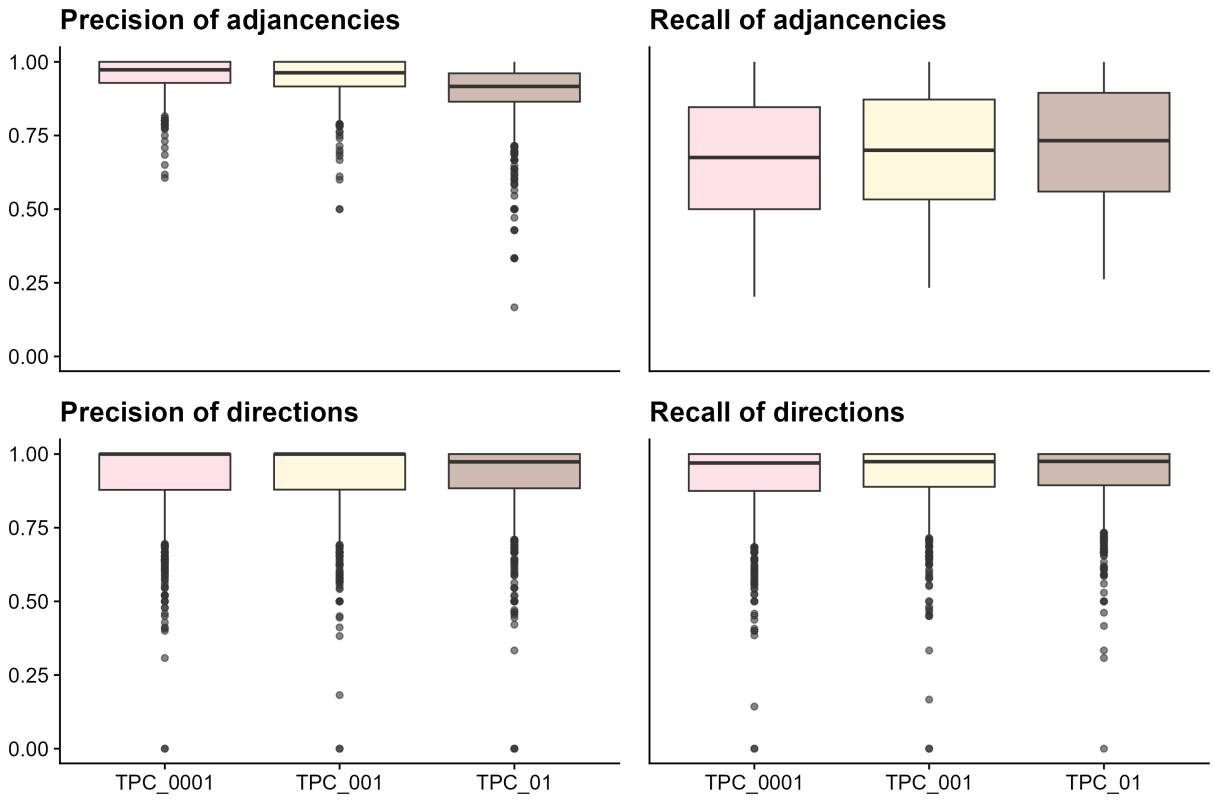


Figure 18: Precision and recall of adjacencies and directions computed for 800 simulations. The methods included are TPC with sparsity level 0.001 (light pink), TPC with sparsity level 0.01 (light yellow) and TPC with sparsity level 0.1 (light brown).

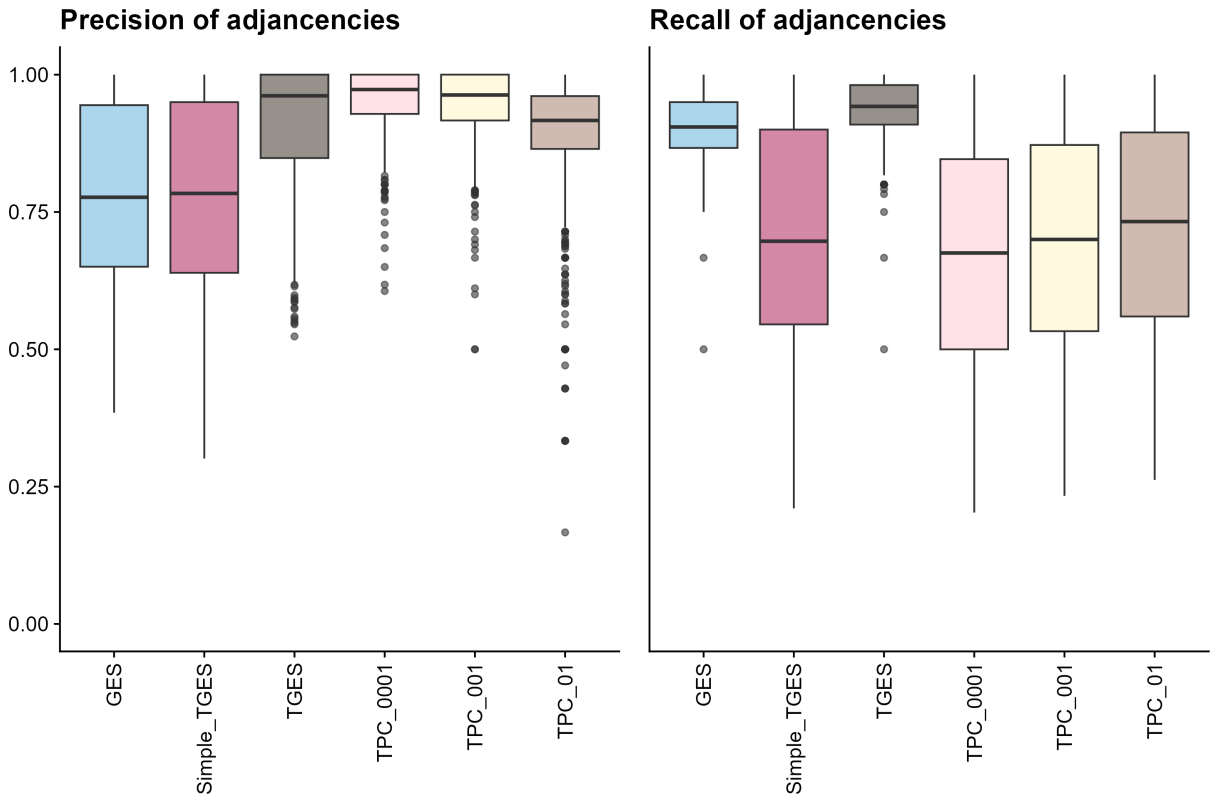


Figure 19: Precision and recall of adjacencies computed for 800 simulations. The methods included are GES (blue), Simple TGES (red), TGES (black), TPC with sparsity level 0.001 (light pink), TPC with sparsity level 0.01 (light yellow) and TPC with sparsity level 0.1 (light brown).

			TGES				Simple TGES				GES				TPC			
		# nodes	7		20		7		20		7		20		7		20	
m	In-tier	Cross-tier\ # tiers	2	5	2	5	2	5	2	5	2	5	2	5	2	5	2	5
30	0.1	0.1	0.67	1.00	0.48	0.65	0.51	0.70	0.43	0.55	0.52	0.70	0.43	0.55	0.98	1.00	0.96	0.94
		0.9	0.27	0.33	0.34	0.28	0.21	0.67	0.33	0.26	0.30	0.67	0.33	0.25	0.85	1.00	0.91	0.68
	0.9	0.1	0.46	0.55	0.37	0.52	0.34	0.47	0.34	0.43	0.38	0.53	0.33	0.43	0.92	0.96	0.95	0.71
		0.9	0.16	0.19	0.29	0.23	0.15	0.27	0.32	0.33	0.24	0.60	0.31	0.31	0.84	0.51	0.97	0.55
1000	0.1	0.1	0.94	0.99	0.89	0.92	0.87	1.00	0.78	0.84	0.88	1.00	0.78	0.86	1.00	1.00	0.95	0.95
		0.9	0.88	0.53	0.46	0.34	0.65	0.40	0.29	0.26	0.65	0.60	0.29	0.25	0.98	0.40	0.66	0.53
	0.9	0.1	0.41	0.94	0.38	0.58	0.37	0.90	0.35	0.44	0.36	0.93	0.35	0.44	0.57	0.97	0.42	0.67
		0.9	0.18	0.40	0.28	0.22	0.22	0.23	0.35	0.36	0.21	0.33	0.35	0.36	0.35	0.69	0.40	0.41

Table 12: For each combination of parameters, the mean of the precision of in-tier directions was evaluated on 100 simulations. Rows are divided among sample size (m), probability of an edge existing between two nodes in the same tier of the data-generating DAG (In-tier), and probability of an edge existing between two nodes in different tiers of the data-generating DAG (Cross-tier). Columns are divided among method used for prediction, number of nodes (#nodes), and number of tiers (#tiers) in the data-generating DAG. Lines and grey scale grading were added for a better overview, rows and columns with red numbers are of questionable interpretation due to very small number of in-tier edges in true graph.

			TGES				Simple TGES				GES				TPC			
		# nodes	7		20		7		20		7		20		7		20	
m	In-tier	Cross-tier\ # tiers	2	5	2	5	2	5	2	5	2	5	2	5	2	5	2	5
30	0.1	0.1	0.85	1.00	0.70	0.74	0.85	1.00	0.70	0.74	0.74	1.00	0.70	0.73	0.64	1.00	0.20	0.35
		0.9	0.83	0.67	0.42	0.39	0.83	0.67	0.43	0.39	0.78	0.67	0.43	0.37	0.73	0.33	0.09	0.16
	0.9	0.1	0.69	0.85	0.39	0.57	0.70	0.80	0.36	0.52	0.61	0.76	0.36	0.51	0.46	0.65	0.03	0.14
		0.9	0.46	0.57	0.32	0.27	0.44	0.60	0.36	0.39	0.36	0.48	0.34	0.35	0.42	0.58	0.01	0.10
1000	0.1	0.1	0.99	0.99	0.94	0.97	0.99	0.97	0.90	0.95	0.95	0.83	0.86	0.92	0.97	0.90	0.92	0.92
		0.9	0.95	0.80	0.51	0.43	0.90	0.73	0.40	0.42	0.90	0.73	0.39	0.42	1.00	0.80	0.67	0.65
	0.9	0.1	0.76	0.99	0.43	0.66	0.75	0.97	0.40	0.55	0.67	0.80	0.40	0.54	0.75	0.96	0.39	0.70
		0.9	0.41	0.69	0.32	0.26	0.46	0.62	0.42	0.44	0.42	0.51	0.41	0.43	0.62	0.93	0.38	0.46

Table 13: For each combination of parameters, the mean of the recall of in-tier directions was evaluated on 100 simulations. Rows are divided among sample size (m), probability of an edge existing between two nodes in the same tier of the data-generating DAG (In-tier), and probability of an edge existing between two nodes in different tiers of the data-generating DAG (Cross-tier). Columns are divided among method used for prediction, number of nodes (#nodes), and number of tiers (#tiers) in the data-generating DAG. Lines and grey scale grading were added for a better overview, rows and columns with red numbers are of questionable interpretation due to very small number of in-tier edges in true graph.

A.5 GES with complete cross-tier edges

Pseudo code for ”GES with complete cross-tier edges”:

Algorithm 5 GES with complete cross-tier edges

Initialize:
Insert all cross-tier directional edges in the DAG $\mathcal{G}$ that is allowed by $\mathcal{K}$
Compute initial score of $\mathcal{G}$.
while There are changes that improve the score **do**
    Forward phase for GES
    Run GES forward phase on PDAG $\mathcal{G}$ but only scoring pair of nodes in same tier.
    Returns a PDAG $\mathcal{G}$
    Backward phase
    Run GES backward phase on entire PDAG $\mathcal{G}$
    Returns PDAG $\mathcal{G}$
    Turning phase
    Run GES turning phase on entire PDAG $\mathcal{G}$
    Returns a PDAG $\mathcal{G}$
end while
Output:
The PDAG $\mathcal{G}$

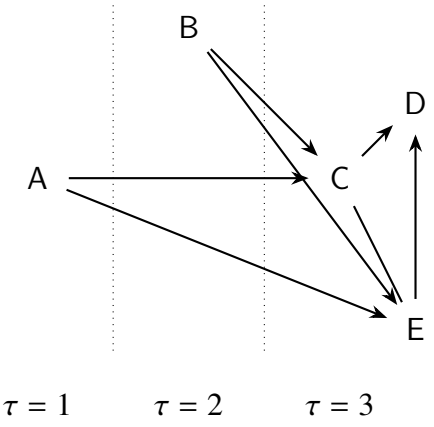


Figure 20: ”GES with complete cross-tier edges” graph prediction of example in Section 3.4.3.