Identification in Tree-shaped Linear Structural Causal Models

Benito van der Zander[+] **Marcel Wienöbst**[+] **Markus Bläser**[*] **Maciej Liśkiewicz**[+]

[+] University of Lübeck, Germany [*] Saarland University, Saarland Informatics Campus, Saarbrücken, Germany

Abstract

Linear structural equation models represent direct causal effects as directed edges and confounding factors as bidirected edges. An open problem is to identify the causal parameters from correlations between the nodes. We investigate models, whose directed component forms a tree, and show that there, besides classical instrumental variables, missing cycles of bidirected edges can be used to identify the model. They can yield systems of quadratic equations that we explicitly solve to obtain one or two solutions for the causal parameters of adjacent directed edges. We show how multiple missing cycles can be combined to obtain a unique solution. This results in an algorithm that can identify instances that previously required approaches based on Gröbner bases, which have doubly-exponential time complexity in the number of structural parameters.

1 INTRODUCTION

Linear structural causal models (SCMs or structural equation models, SEMs) are frequently used to express and analyze the relationships between random variables of interest (Bollen, 1989; Duncan, 1975). Each variable V_i, with $i = 0, \ldots, n$ is assumed to be linearly dependent on the remaining variables and an error term ε_i of normal distribution with zero mean and some covariance matrix $\Omega = (\omega_{ij})$ between the terms:

$$V_i = \sum_j \lambda_{ji} V_j + \varepsilon_i.$$

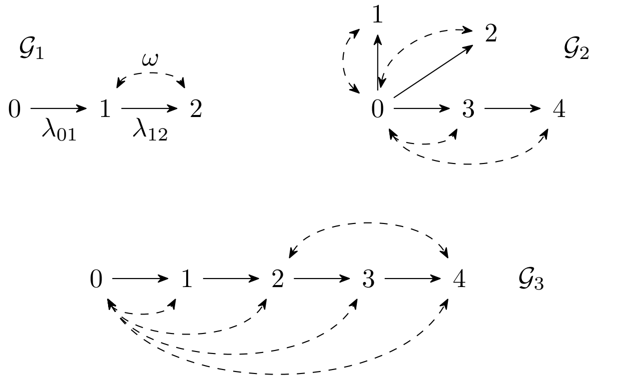


Figure 1: $\mathcal{G}_1$: the classic instrumental variables (IV) model. $\mathcal{G}_2$ is (generically) identifiable by the TSID algorithm (Weihs et al., 2018) and our method TreeID but for which both the half-trek criterion (HTC) and the ACID algorithm (Kumor et al., 2020) fail. Graph $\mathcal{G}_3$ is identifiable by TreeID but not by TSID.

In this paper we consider *recursive models*, i.e. we assume that, for all $j > i$, we have $\lambda_{ji} = 0$. Such a model can be represented as a graph with nodes over the variables. Directed edges represent a linear influence λ_{ji} of a parent node j on its child i. Bidirected edges represent an additional correlation $\omega_{ij} \neq 0$ between random error terms. Given the graph and the weights (also called coefficients or direct causal effects) λ_{ij}, one can calculate the covariances between variables along the paths. For example, in $\mathcal{G}_1$ in Fig. 1, which models random variables V_0, V_1, and V_2, a unit change of V_0 implies a change of λ_{01} of V_1 and a change of $\lambda_{01}\lambda_{12}$ of V_2. The covariance σ_{01} (σ_{02}) between V_0 and V_1 (V_2) is thus λ_{01} ($\lambda_{01}\lambda_{12}$).

Writing the coefficients of all directed edges as an adjacency matrix $\Lambda = (\lambda_{ij})$ and the coefficients of all bidirected edges as an adjacency matrix $\Omega = (\omega_{ij})$, the covariances σ_{ij} between each pair of random variables V_i and V_j can be calculated as matrix $\Sigma = (\sigma_{ij})$:

$$\Sigma = (I - \Lambda)^{-1}\Omega(I - \Lambda)^{-T} \tag{1}$$

Of interest is the reverse problem, the identification and estimation of causal effects. That is, given the graph and a matrix Σ, compute the matrix Λ. The

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151. Copyright 2022 by the author(s).

identification problem, which is the main focus of this paper, asks for a symbolic equation to calculate the coefficients in Λ, the *estimation problem* asks for a numerical solution. Not every coefficient can be identified, yielding the problem to determine which coefficients have solutions.

Identification in linear SCMs has been the subject of a considerable amount of research in the last decades, including the early work in econometrics (Wright, 1928; Fisher, 1966; Bowden and Turkington, 1984) and the pioneering work on the computational aspects of the problem (Pearl, 2009). Most approaches for solving the problem are based on instrumental variables, in which the causal effect is identified as a fraction of two covariances (Wright, 1928; Bowden and Turkington, 1984). For example, in $\mathcal{G}_1$ in Fig. 1, one can calculate $\lambda_{12} = \frac{\lambda_{01}\lambda_{12}}{\lambda_{01}} = \frac{\sigma_{02}}{\sigma_{01}}$. The variable V_0 is then called an instrumental variable (IV). The literature focuses on developing criteria to decide whether a variable is an IV. A more complex criterion – the criterion for a *conditional instrumental variable* (cIV) – considers these correlations of V_0 conditionally on another set of variables (Bowden and Turkington, 1984; Pearl, 2001; van der Zander et al., 2015). Other criteria and methods to identify some coefficients in specific graphs involve instrumental sets (IS) (Brito and Pearl, 2002a; Brito, 2010; Brito and Pearl, 2002b; van der Zander and Liśkiewicz, 2016), half-treks (HTC) (Foygel et al., 2012), auxiliary instrumental variables (aIV) (Chen et al., 2015), determinantal instrumental variables (tsIV) which results in the TSID algorithm (Weihs et al., 2018), instrumental cutsets (Kumor et al., 2019), or auxiliary cutsets which result in the ACID algorithm (Kumor et al., 2020). Some criteria lead to polynomial-time algorithms (ACID). For other criteria, e.g. cIV and tsIV, the decision problem, if the criterion is satisfied in a given graph, is NP-complete (van der Zander et al., 2015; van der Zander and Liśkiewicz, 2016; Kumor et al., 2019).

A drawback to all criteria listed above is that they are not complete, i.e. not applicable to every graph. They also only decide whether there exists exactly one solution for a coefficient.

An alternative approach is to expand Eq. (1) to a system of polynomial equations and solve it using a computer algebra system (CAS), which usually employs Gröbner bases (García-Puente et al., 2010; Foygel et al., 2012). This gives a complete solution for any solvable equation system. However, Gröbner base algorithms have a doubly exponential runtime and are EXPSPACE-complete (Mayr and Meyer, 1982). Thus, they are often too slow to be used in practice. García-Puente et al. (2010) note the runtime varies between seconds and 75 days for graphs with four nodes.

Our results. We investigate the identification problem in linear SCMs on graphs whose directed component is a tree and whose bidirected component can be arbitrary. Figure 1 shows example causal structures, which are modeled as tree graphs.

First, we show that if a node i is not connected to the root node with a bidirected edge, the root node can be used as a classic instrumental variable to identify the incoming edge to i. Then we investigate the subgraph of nodes that are each connected to the root node with a bidirected edge. We show that, if there is a missing cycle, such that none of the bidirected edges $i_1 \leftrightarrow i_2 \leftrightarrow i_3 \leftrightarrow \ldots \leftrightarrow i_1$ exists in the graph, it yields an equation system that can lead to a solution of all incoming directed edges to the involved nodes. We describe how to reduce this equation system to a single quadratic equation in a single variable, which can then identify one of the incoming edges. As a quadratic equation, it can yield exactly one, exactly two, or infinitely many solutions. How many solutions exist for a certain graph can be symbolically determined using Polynomial Identity Testing (PIT). For example, a graph with directed component $0 \to 1 \to 2 \to 3$ and bidirected edges $0 \leftrightarrow i$ for $i = 1, 2, 3$ does not contain the bidirected cycle $1 \leftrightarrow 2 \leftrightarrow 3 \leftrightarrow 1$, so our approach returns exactly two solutions for the identification of $\lambda_{01}, \lambda_{12}, \lambda_{23}$.

This results in an algorithm, we call TreeID, that can identify instances that previously required the Gröbner bases approach and for which the state-of-the-art methods, such as HTC, TSID, and ACID fail.

We have performed the necessary calculations on a large number of graphs, with a special focus on graphs whose directed edges form a single directed path and where the root node is connected to all other nodes with bidirected edges. If the bidirected component is complete except for a missing cycle of a length between five and ten edges, there are always exactly two solutions.

Identification of Tree Graphs with the State-of-the-Art-Methods. The algorithm ACID (Kumor et al., 2020), which subsumes previous state-of-the-art methods, including cIV, IS, aIV, and HTC, as well as the TSID approach (Weihs et al., 2018) belong currently to the most prominent and powerful methods for identification of structural coefficients in linear causal models. They are significantly more efficient compared to the general Gröbner bases approach.

For tree graphs, however, more sophisticated criteria are often either not applicable or not advantageous compared to simpler ones. This can be most easily seen in the case of ISs. As all nodes have in-degree 1, it is not possible to identify *two or more* incoming edges at once.

We show that many of the previous criteria and algorithms, including ACID, collapse to the use of auxiliary instrumental variables (aIVs). This result shows that making progress beyond simple rules such as aIV on tree graphs appears to be quite challenging. One explanation could of course be that, e. g., the ACID algorithm is already powerful enough to identify such simple models. However, this turns out not to be the case. Indeed, there is a large number of tree graphs not identifiable in this manner as is shown exemplarily in Fig. 1: Graph $\mathcal{G}_2$, which is taken from Fig. 3c in Foygel et al. (2012), is generically identifiable by TSID and our algorithm TreeID but for which both the HTC and ACID fail to identify coefficients.

Several of the tree graphs, like $\mathcal{G}_1$ and $\mathcal{G}_2$ in Fig. 1, can be identified with the TSID algorithm implementing the tsIV criterion (Weihs et al., 2018). This criterion, however, still leaves a significant number of tree graphs unidentified. For an example, see $\mathcal{G}_3$ in Fig. 1, which is identifiable by TreeID. Moreover, the tsIV criterion has the drawback that it is likely not efficiently testable as it was shown to be NP-hard (Kumor et al., 2020).

Our algorithm TreeID is an entirely new approach for identification independent of the instrumental variable framework.

2 PRELIMINARIES

We consider mixed acyclic graphs $\mathcal{G} = (V, D, B)$ with $n + 1$ nodes $V = \{0, 1, \ldots, n\}$, directed edges $i \to j$ in D and bidirected edges $i \leftrightarrow j$ in B. By acyclicity, we mean that $\mathcal{G} = (V, D)$ restricted to directed edges is a directed acyclic graph (DAG).

For a given graph $\mathcal{G}$, the *identification problem* asks to find, for each parameter λ_{xy}, with $x \to y \in D$, an expression to calculate λ_{xy} that only depends on the entries of the matrix Σ, such that the calculated λ_{xy} uniquely satisfies Eq. (1). Practically, there are initial values of Λ and Ω, from which a matrix Σ is calculated. Using the expression for λ_{xy}, new matrices Λ' and Ω' can be calculated and it needs to hold

$$(I - \Lambda')^{-1}\Omega'(I - \Lambda')^{-T} = \Sigma = (I - \Lambda)^{-1}\Omega(I - \Lambda)^{-T}.$$

We consider the generic version of the problem, which only requires that the expressions are valid almost everywhere, i.e. the Lebesgue measure of the set of initial parameters for which the expressions are not valid is zero. E.g. it is allowed to return solutions like $\lambda = \frac{\sigma_{zy}}{\sigma_{zx}}$ even though they are not valid for $\sigma_{zx} = 0$. Since any algebraic subset has Lebesgue measure zero, we can assume that any polynomial over elements of Λ, Ω evaluates to zero if and only if it is the zero polynomial. By $|ID_{\mathcal{G}}(\lambda_{xy})|$, we denote the degree of identifiability of edge λ_{xy}. $|ID_{\mathcal{G}}(\lambda_{xy})| = 1$ if λ_{xy} is *uniquely identifiable*, $|ID_{\mathcal{G}}(\lambda_{xy})| = \infty$ if *not identifiable*, and $1 < |ID_{\mathcal{G}}(\lambda_{xy})| < \infty$ if identifiable with more than one solution, e.g. $|ID_{\mathcal{G}}(\lambda_{xy})| = 2$ means *2-identifiable*.

A *path* in a graph $\mathcal{G}$ is a node sequence $i_1, \ldots, i_{\ell+1}$ such that all successive nodes i_k, i_{k+1}, with $1 \le k \le \ell$, are connected by a directed edge. Then i_1 is called the *start node* and $i_{\ell+1}$ the *end node* of the path. We use the terms *child*, *parent*, *ancestor*, *descendant*, and *sibling* to describe node relationships in graphs in the same way as Pearl (2009); in this convention, every node is an ancestor (but not a parent) and a descendant (but not a child) of itself. For a node i, we denote by $An(i)$ the set of all ancestors of i.

A *trek* τ in $\mathcal{G}$ from source i to target j is a path from i to j whose consecutive edges do not have any colliding arrowheads, i.e. τ a path of one of the two following forms $i \leftarrow i_1 \leftarrow \ldots \leftarrow u \leftrightarrow v \to j_1 \to \ldots \to j$ where node i can coincide with u or j can coincide with v, or $i \leftarrow i_1 \leftarrow \ldots \leftarrow u \to j_1 \to \ldots \to j$ where either i or j can coincide with u. Define the *trek monomials* $M(\tau)$ as follows. For τ of the first form, define $M(\tau) = \omega_{uv} \prod_{x \to y \in \tau} \lambda_{xy}$ and, for τ of the second form, define $M(\tau) = \omega_{uu} \prod_{x \to y \in \tau} \lambda_{xy}$. Then, the following *trek rule* (Wright, 1921, 1934) expresses the covariance matrix (1) as a summation over all treks

$$\sigma_{ij} = \sum_{\tau \text{ trek from } i \text{ to } j} M(\tau). \tag{2}$$

In this paper, we restrict ourselves to *tree graph models*, i.e. assume that $\mathcal{G} = (V, D)$ is a directed tree, which has exactly one node, called *root*, whose incoming degree is zero and all other nodes have incoming degree one. The root node is labeled 0. For each node i, with $i > 0$ and its (unique) parent p, the coefficient of the incoming edge $p \to i$ is denoted as λ_{pi}, also written sometimes as λ_i for short.

Finally, we recall some concepts generalizing IVs, which are relevant to our paper. The idea of auxiliary variables aIV (Chen et al., 2017) is to utilize already identified direct effects for further identification. Assume for variable y, the incoming edge $x \to y$ has been identified. Then, one can create the variable $y^* = y - \lambda_{xy}x$, i.e. subtract out the identified direct effect. The resulting auxiliary variable y^* acts as if there is no edge $x \to y^*$, which enables further identification. The aIV criterion identifies edges by the instrumental variable criterion and creates corresponding auxiliary variables until no further edge can be identified.

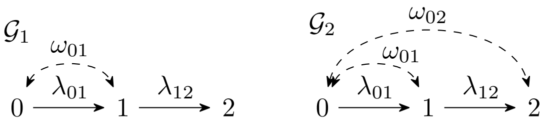


Figure 2: The polynomial $\lambda_{12}\sigma_{10} - \sigma_{20}$ vanishes in the model $\mathcal{G}_1$ but it is a nonzero-polynomial in $\mathcal{G}_2$.

3 RELATING SYMBOLIC EQUATIONS WITH POLYNOMIAL IDENTITY TESTING

To find the solutions for λ_{ij} in a given tree graph $\mathcal{G} = (V, D, B)$, our algorithm handles multivariate functions involving polynomials over λ_{ij}, with $i \to j \in D$, and σ_{ij}, with $0 \le i, j \le n$. An important task that the algorithm has to cope with during the computation is to verify whether a formula F for a parameter λ satisfies an equation involving expressions over σ_{ij}. Another problem is to check if a given F is a zero-function. E.g., is $\lambda_{12}\sigma_{10} - \sigma_{20}$ the zero-polynomial? One can easily see that for the graph $\mathcal{G}_1$ in Fig. 2 this is the case but for $\mathcal{G}_2$ not.

Below we show that these tasks can be reduced to Polynomial Identity Testing (PIT) and in consequence solved efficiently using the well-known approach based on the Schwartz-Zippel lemma[1]. The lemma states the probability of a non-zero polynomial evaluating to zero at random variable values is negligible.

Lemma 1 (Schwartz-Zippel)**.** *Let $p(x_1, \ldots, x_n)$ be a non-zero polynomial of total degree $\le d$ over a field $\mathbb{F}$. Let $S \subseteq \mathbb{F}$ be a finite set and let $a_1, \ldots, a_n$ be selected at random independently and uniformly from S. Then* $\Pr[p(a_1, \ldots, a_m) \neq 0] \ge 1 - d/|S|$.

Our algorithm represents formulas in a form, we call *fractional affine square-root terms of polynomials* (FASTP); We define it as $\frac{p+q\sqrt{s}}{r+t\sqrt{s}}$, where p, q, r, s, t are multivariate polynomials. In particular, the algorithm represents parameters λ as FASTPs with p, q, r, s, t over σ_{ij}.

Definition 1. *Let $\mathcal{G} = (V, D, B)$ over $V = \{0, \ldots, n\}$ be an arbitrary mixed graph. For a FASTP F over λ_{ij}, with $i \to j \in D$, and σ_{ij}, with $0 \le i, j \le n$, let $[F]_{\mathcal{G}}$ be the substitution of all σ_{ij} with terms in $\lambda_{ij}, \omega_{ij}$ according to the trek rule* (2)*. Thus, assuming we get no division by a zero-polynomial, $[F]_{\mathcal{G}}$ is a FASTP over λ_{ij}, with $i \to j \in D$, and ω_{ij}, with $i \leftrightarrow j \in B$.*

For example, for the polynomial $F = \lambda_{12}\sigma_{10} - \sigma_{20}$ above and the models $\mathcal{G}_1$ and $\mathcal{G}_2$ in Fig. 2, we have $[F]_{\mathcal{G}_1} = \lambda_{12}(\omega_{01} + \lambda_{01}\omega_{00}) - \lambda_{12}\omega_{01} - \lambda_{12}\lambda_{01}\omega_{00}$ and $[F]_{\mathcal{G}_2} = \lambda_{12}(\omega_{01} + \lambda_{01}\omega_{00}) - \lambda_{12}\omega_{01} - \lambda_{12}\lambda_{01}\omega_{00} - \omega_{02}$. Thus, $[F]_{\mathcal{G}_1}$ is the zero-polynomial but $[F]_{\mathcal{G}_2} = -\omega_{02}$ not. This implies that F vanishes when considering model $\mathcal{G}_1$ but it is a nonzero-polynomial in $\mathcal{G}_2$.

Now, we are ready to show, that testing if λ represented as a FASTP satisfies a specific equation can be reduced to PIT.

Lemma 2. *For a given FASTP $\lambda = \frac{p+q\sqrt{s}}{r+t\sqrt{s}}$ and polynomials a, b, c, one can verify – up to a sign – whether $\left[a\lambda^2 + b\lambda + c\right]_{\mathcal{G}} \equiv 0$ holds using PIT.*

Proof. The equality $(a(q\sqrt{s} + p)^2)/(t\sqrt{s} + r)^2 + (b(q\sqrt{s} + p))/(t\sqrt{s} + r) + c \equiv 0$ can be expressed as: $cst^2 + \sqrt{s}(2crt + bpt + bqr + 2apq) + bqst + aq^2s + cr^2 + bpr + ap^2 \equiv 0$.

We distinguish two cases: If s is a perfect square (in the ring of polynomials), that is, there is a polynomial ς such that $\varsigma^2 = s$, then the above equation is an instance of polynomial identity testing. If s is not a perfect square, then the left-hand side of the equation above can only be identically zero if $2crt + bpt + bqr + 2apq \equiv 0$ and $cst^2 + bqst + aq^2s + cr^2 + bpr + ap^2 \equiv 0$ (otherwise, $\sqrt{s}$ would be an element of the rational function field). These are two instances of PIT.

For this approach, we need to be able to check whether s is a perfect square and if so, compute an arithmetic circuit for ς. Brent and Kung (1978) propose to use Newton iteration to approximate the square root of a polynomial by a power series. If the polynomial is a perfect square, then this approximation is exact. This even works for multivariate polynomials given by arithmetic circuits, some of the details are spelled out in Bläser et al. (2017). Given a circuit C for s, we use this algorithm to compute a circuit D for a candidate square root $\hat{\varsigma}$. We can now use PIT to check whether $\hat{\varsigma}^2 = s$. If yes, then D is a circuit for $\sqrt{s}$. In the no case, s is not a perfect square. □

If s is a perfect square, then there are two square roots, namely ς and $-\varsigma$. There is a priori no canonical way to distinguish these two[2]. If s is a perfect square, then the output of the algorithm is called $\sqrt{s}$ and the other root is $-\sqrt{s}$.

In our algorithm, the degree of all involved polynomials is polynomially bounded, which ensures that the resulting PITs can be solved effectively.

[1] Schwartz (1980); Zippel (1979); DeMillo and Lipton (1978)

[2] While implementing our algorithm, we have noticed that the most practical way of deciding this and the equivalence of Lemma 2 is to choose random values for variables λ_{ij} and ω_{ij}, compute σ_{ij} using Equation (2), and just evaluate the polynomials with arbitrary precision arithmetic.

4 BASIC EQUATIONS AND PRELIMINARY IDENTIFICATIONS

In this section, we examine the covariances σ_{ij} in a tree model $\mathcal{G}$.

Let, for two nodes s and t connected by a directed path π, the function $L(s,t)$ be defined as follows: $L(s,t) = \prod_{x\to y\in\pi}\lambda_{xy}$, with $L(s,s)=1$.

Lemma 3. *For a given tree graph $\mathcal{G}=(V,D,B)$ and for any two nodes i,j in V we have: $[\sigma_{ij}]_{\mathcal{G}} = \sum_{s\in An(i)}\sum_{t\in An(j)}\omega_{st}L(s,i)L(t,j)$*

Thus, for a tree graph $\mathcal{G}$ and a multivariate polynomial F over λ_{ij} and σ_{ij}, we get $[F]_{\mathcal{G}}$ by the substitution of all σ_{ij} according to the equation in Lemma 3.

The two lemmas below can be proved easily and are special cases of well-known results, see e.g. (Drton, 2018).

Lemma 4. *Assume $\mathcal{G}$ is a tree graph and let i,j be two different nodes. Then, for all $i,j>0$ and for (unique) parents p,q with edges $p\to i, q\to j$, we have*

$$\omega_{ij} = [\lambda_{pi}\lambda_{qj}\sigma_{pq} - \lambda_{pi}\sigma_{pj} - \lambda_{qj}\sigma_{iq} + \sigma_{ij}]_{\mathcal{G}}\,.$$

Moreover, for all $j>0$ and the parent q with edge $q\to j$, it is true

$$\omega_{0j} = [\sigma_{0j} - \lambda_{qj}\sigma_{0q}]_{\mathcal{G}}\,.$$

Based on Lemma 4, we analyze first the structure of equations of the system $(I-\Lambda)\Sigma(I-\Lambda)^T = \Omega$.

Lemma 5. *An edge λ_{xy} is identifiable in a tree graph $\mathcal{G}$, if (and only if) the system of the equations*

- *$\lambda_{pi}\lambda_{qj}\sigma_{pq} - \lambda_{pi}\sigma_{pj} - \lambda_{qj}\sigma_{iq} + \sigma_{ij} = 0$, for all missing edges $i\leftrightarrow j$, where p, resp. q, are parents of i, resp. j, and*
- *$\sigma_{0i} - \lambda_{pi}\sigma_{0p} = 0$, for all missing edges $0\leftrightarrow i$, where p is a parent of i,*

has a (unique) solution for λ_{xy}. Moreover, the number of generic solutions for λ_{xy} is equal to $|ID_{\mathcal{G}}(\lambda_{xy})|$.

From Lemma 5, two obvious ways of identifying certain edges emerge.

Corollary 1. *If the edge $0\leftrightarrow i$ is missing for $i>0$ with parent p, then λ_{pi} is identified as $\lambda_{pi} = \sigma_{0i}/\sigma_{0p}$.*

Corollary 2. *If the edge $i\leftrightarrow j$ is missing for i with parent p and j with parent q, λ_{pi} is identified, and $[(\lambda_{pi}\sigma_{pq}-\sigma_{iq})]_{\mathcal{G}} \not\equiv 0$, then λ_{qj} is identified as $\lambda_{qj} = (\lambda_{pi}\sigma_{pj}-\sigma_{ij})/(\lambda_{pi}\sigma_{pq}-\sigma_{iq})$.*

Below we show, that in many tree graphs the inequality required in the corollary above is true.

Lemma 6 (Propagation)**.** *Let $\mathcal{G}=(V,D,B)$ be a tree graph, and let i,j, with $i\leftrightarrow j\notin B$, be two different nodes with parents $p\to i$ and $q\to j$. Then $[(\lambda_{pi}\sigma_{pq}-\sigma_{iq})]_{\mathcal{G}}\not\equiv 0$ if and only if there is a trek from i and q in $\mathcal{G}\setminus\{p\to i\}$. In particular, the polynomial is non-zero in $\mathcal{G}$ if $\mathcal{G}$ contains the bidirected edge $0\leftrightarrow i$ or if q is a descendant of i in $\mathcal{G}$.*

Proof. Assume first that q is a descendant of i in $\mathcal{G}$. Then both there is a trek from i and q in $\mathcal{G}\setminus\{p\to i\}$ as well as $[(\lambda_{pi}\sigma_{pq}-\sigma_{iq})]_{\mathcal{G}}\not\equiv 0$ since, for a trek τ' from i and q in $\mathcal{G}\setminus\{p\to i\}$, the summation $\sigma_{iq} = \sum_{\tau \text{ trek from } i \text{ to } q} M(\tau)$ includes a term $M(\tau')$ which does not involve λ_{pi}.

If q is not a descendant of i in $\mathcal{G}$, then we can express σ_{iq} as

$$\sigma_{iq} = \lambda_{pi}\sigma_{pq} + \sum_{\tau \text{ trek from } i \text{ to } q \text{ without } p\to i} M(\tau).$$

If there is a trek from i and q in $\mathcal{G}\setminus\{p\to i\}$, then the sum on the right-hand side does not vanish and $[(\lambda_{pi}\sigma_{pq}-\sigma_{iq})]_{\mathcal{G}}\not\equiv 0$. On the other hand, if the polynomial is non-zero, then

$$\sum_{\tau \text{ trek from } i \text{ to } q \text{ without } p\to i} M(\tau)\not\equiv 0.$$

This means that in $\mathcal{G}$ there exists a trek τ' from i to q without $p\to i$. □

Hence, we use the two Corollaries 1 and 2 for a simple preliminary identification step by checking for which nodes Corollary 1 applies and recursively utilizing Corollary 2 and Lemma 6 whenever a new edge is identified. In particular, we will refer to the recursive strategy as *propagation*.

The preliminary identification of edges in this manner is a simple and efficient implementation of the aIV strategy:

Proposition 1. *Every edge in a tree graph identified by aIV is identified during preliminary identification.*

Moreover, we show that, for tree graphs, preliminary identification is at least as effective as the state-of-the-art polynomial-time algorithm ACID.

Proposition 2. *Every edge in a tree graph identified by the ACID algorithm is identified during preliminary identification.*

In the following sections, we derive an entirely new approach to direct effect identification based on missing cycles of bidirected edges, able to identify even further parameters.

5 MISSING CYCLE EQUATIONS

In this section, we show how a missing cycle of bidirected edges can yield an identification of directed edges that point at the nodes of the cycle:

Definition 2. *Let $v_1, \ldots, v_k > 0$ be a missing cycle with parents $p_i \to v_i$. Let $v_{k+1} = v_1$ and $p_{k+1} = p_1$. Let $L = \lceil \log_2 k \rceil + 1$.*

Define the polynomials $a_i^{(l)}, b_i^{(l)}, c_i^{(l)}, d_i^{(l)}$, for $l = 1, \ldots, L$, recursively as follows

$$a_i^{(l+1)} = \begin{cases} \sigma_{p_i,p_{i+1}} & l = 0 \\ a_{2i-1}^{(l)} & \lceil \frac{k}{2^{l-1}} \rceil \textit{ is odd} \wedge i = \lceil \frac{k}{2^l} \rceil \\ \det \begin{pmatrix} a_{2i-1}^{(l)} & a_{2i}^{(l)} \\ b_{2i-1}^{(l)} & c_{2i}^{(l)} \end{pmatrix} & \textit{else,} \end{cases}$$

$$b_i^{(l+1)} = \begin{cases} -\sigma_{p_i,v_{i+1}} & l = 0 \\ b_{2i-1}^{(l)} & \lceil \frac{k}{2^{l-1}} \rceil \textit{ is odd} \wedge i = \lceil \frac{k}{2^l} \rceil \\ \det \begin{pmatrix} a_{2i-1}^{(l)} & b_{2i}^{(l)} \\ b_{2i-1}^{(l)} & d_{2i}^{(l)} \end{pmatrix} & \textit{else,} \end{cases}$$

$$c_i^{(l+1)} = \begin{cases} -\sigma_{v_i,p_{i+1}} & l = 0 \\ c_{2i-1}^{(l)} & \lceil \frac{k}{2^{l-1}} \rceil \textit{ is odd} \wedge i = \lceil \frac{k}{2^l} \rceil \\ \det \begin{pmatrix} c_{2i-1}^{(l)} & a_{2i}^{(l)} \\ d_{2i-1}^{(l)} & c_{2i}^{(l)} \end{pmatrix} & \textit{else,} \end{cases}$$

$$d_i^{(l+1)} = \begin{cases} \sigma_{v_i,v_{i+1}} & l = 0 \\ d_{2i-1}^{(l)} & \lceil \frac{k}{2^{l-1}} \rceil \textit{ is odd} \wedge i = \lceil \frac{k}{2^l} \rceil \\ \det \begin{pmatrix} c_{2i-1}^{(l)} & b_{2i}^{(l)} \\ d_{2i-1}^{(l)} & d_{2i}^{(l)} \end{pmatrix} & \textit{else.} \end{cases}$$

A missing cycle encodes a quadratic equation for each incoming edge that can yield two possible solutions:

Theorem 1. *Let $\mathcal{G} = (V, D, B)$ be a tree graph and assume there is a cycle $v_1, \ldots, v_k > 0$, such that each edge $v_i \leftrightarrow v_{i+1}$, with $v_{k+1} = v_1$, is missing. Then the path coefficient λ_{v_1} satisfies the equation*

$$a_1^{(L)} \lambda_{v_1}^2 + (b_1^{(L)} + c_1^{(L)}) \lambda_{v_1} + d_1^{(L)} = 0 \qquad (3)$$

where a, b, c, d are calculated as in Definition 2.

Lemma 7. *In the following cases, the equation* (3) *of Theorem 1 has one or two solutions:*

If $\left[a_1^{(L)}\right]_{\mathcal{G}} \equiv 0 \wedge \left[b_1^{(L)} + c_1^{(L)}\right]_{\mathcal{G}} \not\equiv 0$, then $|ID_{\mathcal{G}}(\lambda_{v_1})| = 1$.

If $\left[a_1^{(L)}\right]_{\mathcal{G}} \not\equiv 0 \wedge \left[(b_1^{(L)} + c_1^{(L)})^2 - 4a_1^{(L)} d_1^{(L)}\right]_{\mathcal{G}} \not\equiv 0$, then $|ID_{\mathcal{G}}(\lambda_{v_1})| \leq 2$.

If $\left[a_1^{(L)}\right]_{\mathcal{G}} \not\equiv 0 \wedge \left[(b_1^{(L)} + c_1^{(L)})^2 - 4a_1^{(L)} d_1^{(L)}\right]_{\mathcal{G}} \equiv 0$, then $|ID_{\mathcal{G}}(\lambda_{v_1})| = 1$.

Proof. In the first case, the equation becomes linear and has a solution $\lambda_{v_1} = \frac{-d_1^{(L)}}{b_1^{(L)} + c_1^{(L)}}$.

If a solution exists, then it will always be real-valued. Therefore, the polynomial $\left[(b_1^{(L)} + c_1^{(L)})^2 - 4a_1^{(L)} d_1^{(L)}\right]_{\mathcal{G}}$ will always be non-negative. If it is nonzero, then there are two solutions $\lambda_{v_1} = \frac{-(b_1^{(L)} + c_1^{(L)}) \pm \sqrt{(b_1^{(L)} + c_1^{(L)})^2 - 4a_1^{(L)} d_1^{(L)}}}{2a_1^{(L)}}$.

If the square root is zero, then there is effectively only one solution. □

If one edge into a missing cycle is identifiable, all other edges into this missing cycle are also identifiable. From equation $\lambda_{pi}\lambda_{qj}\sigma_{pq} - \lambda_{pi}\sigma_{pj} - \lambda_{qj}\sigma_{iq} + \sigma_{ij} = 0$, it follows $\lambda_{qj} = (\lambda_{pi}\sigma_{pj} - \sigma_{ij})/(\lambda_{pi}\sigma_{pq} - \sigma_{iq})$, so knowing one edge λ_{pi}, one can usually derive the other edges. However, this might not always be possible since $[\lambda_{pi}\sigma_{pq} - \sigma_{iq}]_{\mathcal{G}} = 0$ might occur.

6 THE ALGORITHM

In this section, we present an algorithm, called TreeID, to identify parameters in tree models $\mathcal{G} = (V, D, B)$. We assume $V = \{0, \ldots, n\}$ such that the nodes are numbered in topological order, i.e. if $i \in An(j)$, then $i \leq j$. Thus, in particular, 0 is the root node.

The algorithm, presented as Algorithm 1, uses the array ID$[1, \ldots, n]$ to store the solutions for parameters λ_i for edges $p \to i \in D$, as functions over σ_{jk}. Initially, all ID$[i] = \emptyset$ meaning the parameter is not-identified. At the end of the algorithm, if —ID$[i]| = 1$, then λ_i is identifiable and given by the formula in ID$[i]$. If —ID$[i]| = 2$, then λ_i is identifiable by at least one of the solutions given in ID$[i]$. During its work, the algorithm represents the formulas in ID$[i]$ in the FASTP form $\frac{p+q\sqrt{s}}{r+t\sqrt{s}}$ where p, q, r, s, t are polynomials over σ_{jk}.

TreeID starts with the identification of λ_i for each i, such that $0 \leftrightarrow i \notin B$. To this aim node 0 is used as an instrumental variable (Corollary 1). Based on Corollary 2, the identification for λ_i is "propagated" (recursively) to identify parameters λ_j, with $i \leftrightarrow j \notin B$, $p \to i, q \to j \in D$, as $\lambda_j = (\lambda_i\sigma_{pj} - \sigma_{ij})/(\lambda_i\sigma_{pq} - \sigma_{iq})$, if the function in the denominator is non-zero.

The main part of the algorithm identifies the parameters λ_i, which have not been recognized as identifiable in the initial phase. To this aim, for each such i, TreeID proceeds as follows: For every "missing" cycle $v_1 = i, v_2, \ldots, v_k > 0$ including i, i.e. for a sequence of nodes such that $v_j \leftrightarrow v_{j+1} \notin B$ for all $j = 1, \ldots k$, with $v_{k+1} = v_1$, the algorithm computes a quadratic equation $a\lambda_i^2 + b\lambda_i + c = 0$ using Theorem 1. If both $[a]_{\mathcal{G}} \equiv 0$

and $[b]_{\mathcal{G}} \equiv 0$, then all λ_i satisfy the equation and thus the algorithm skips the cycle. If only $[a]_{\mathcal{G}} \equiv 0$, the equation has exactly one solution, which is stored in ID[i]. Otherwise, if λ_i is not yet identified (ID[i] $= \emptyset$), the algorithm using Lemma 7 computes one or two solutions for λ_i; Otherwise, it updates the solutions ID[i] calculated so far, by removing from the set such λ's that do not satisfy the equation $a\lambda_i^2 + b\lambda_i + c = 0$. Finally, similarly as in the initial phase, the identifications for λ_i are propagated to compute formulas for λ_j, with $i \leftrightarrow j \notin B$. This proves the following:

Theorem 2. *The identification algorithm TreeID is sound for tree graphs, that is, for a given tree $\mathcal{G} = (V, D, B)$, with $V = \{0, \ldots, n\}$, if it returns i and* ID[i], *then λ_i is identifiable and given by the formula. Additionally, if at the end of the algorithm* —ID[i]| $= 2$, *then λ_i is identifiable with at least one of the solutions given in* ID[i].

Algorithm 1: TreeID

```
input: tree graph $\mathcal{G} = (V = \{0, \ldots, n\}, D, B)$
output: a set of identifiable structural parameters

  function SolveEquation()
    input: $a\lambda^2 + b\lambda + c = 0$

    if $[b^2 - 4ac]_{\mathcal{G}} \equiv 0$: return $\{-b/2a\}$
    $s \leftarrow \sqrt{b^2 - 4ac}$
    return $\{(-b - s)/2a, (-b + s)/2a\}$

  function Propagate()    //use Corollary 2
    input: $i$
    $p \leftarrow Pa(i)$
    for each $i \leftrightarrow j \notin B$:
      if $0 < |\mathrm{ID}[j]| \leq |\mathrm{ID}[i]|$: continue
      $q \leftarrow Pa(j)$
      if $\exists \lambda \in \mathtt{ID}[i]\ s.t. [(\lambda\sigma_{pq} - \sigma_{iq})]_{\mathcal{G}} \equiv 0$: continue
      $\mathrm{ID}[j] \leftarrow \{(\lambda\sigma_{pj} - \sigma_{ij})/(\lambda\sigma_{pq} - \sigma_{iq}) \mid \lambda \in \mathtt{ID}[i]\}$
      Propagate($j$)

for $i \leftarrow 1 \ldots n$:
  $\mathrm{ID}[i] \leftarrow \emptyset$ //mark all nodes as not identified

for $i \leftarrow 1 \ldots n$:
  if $0 \leftrightarrow i \notin B$:
    $\mathrm{ID}[i] \leftarrow \{\sigma_{0i}/\sigma_{0p}\}$ //use Corollary 1
    Propagate($i$)

for $i \leftarrow 1 \ldots n$:
  if $|\mathrm{ID}[i]| = 1$: continue
  for each missing cycle involving node $i$:
    Use Thm. 1 to get a quadratic equation
           $a\lambda_i^2 + b\lambda_i + c = 0$
    if $[a]_{\mathcal{G}} \equiv 0$ and $[b]_{\mathcal{G}} \equiv 0$: continue

    if $[a]_{\mathcal{G}} \equiv 0$:
      $\mathrm{ID}[i] \leftarrow \{-c/b\}$
    else if $\mathrm{ID}[i] = \emptyset$:
      $\mathrm{ID}[i] \leftarrow$ SolveEquation($a\lambda_i^2 + b\lambda_i + c = 0$)
    else
      $\mathrm{ID}[i] \leftarrow \{\lambda \in \mathtt{ID}[i] \mid [a\lambda^2 + b\lambda + c]_{\mathcal{G}} \equiv 0\}$

    if $|\mathrm{ID}[i]| = 1$: break
  Propagate($i$)

for $i \leftarrow 1 \ldots n$:
  if —ID[$i$]| $= 1$: return $i$, ID[$i$]
```

The tests $[F]_{\mathcal{G}} \equiv 0$ for FASTPs F over σ_{jk} can be reduced to PITs according to Lemma 2. The standard algorithm for deciding PIT runs in randomized polynomial time using Lemma 1. It is a blackbox algorithm, i.e. it does not require a representation of the polynomial, only the evaluated value of the polynomial. Although the calculation of the FASTP in the propagation step can double the size of the polynomials, TreeID can store the constant size equation $\lambda_j = (\lambda_i\sigma_{pj} - \sigma_{ij})/(\lambda_i\sigma_{pq} - \sigma_{iq})$ directly without expanding λ_i. For PIT, we then evaluate all λ_i, λ_j recursively, storing the value of the four polynomials in the FASTP separately. This gives:

Proposition 3. *For a given tree graph $\mathcal{G}$, the running time of TreeID algorithm is in $O(p(n) \cdot mc_{\mathcal{G}})$ randomized time, where $p(n)$ is a polynomial for solving PIT and $mc_{\mathcal{G}}$ denotes the number of missing (bidirectional) cycles in $\mathcal{G}$.*

One can also see that the algorithm runs in polynomial space (PSPACE).

Proposition 4. *If λ_i is identifiable with the ACID algorithm (that covers cAV, IC, qAVS criteria), then it is identifiable with the TreeID algorithm.*

7 EXAMPLES

In this section we first explain how the algorithm TreeID identifies the graphs in Fig. 1. Next, we show how our algorithm works on instances considered in Weihs et al. (2018) which are *unidentifiable* by HTC and the TSID algorithm. Finally, we discuss path graphs – as a special case of tree graphs, whose bidirected component is complete except for exactly one missing cycle.

7.1 Models in Figure 1

In the classic IV model $\mathcal{G}_1$, the root node has no bidirected edges. Thus TreeID immediately identifies all causal effects as $\lambda_1 = \frac{\sigma_{01}}{\sigma_{00}}$ and $\lambda_2 = \frac{\sigma_{02}}{\sigma_{01}}$ using Corollary 1.

In $\mathcal{G}_2$, the root node is connected to all other nodes by bidirected edges, so Corollary 1 cannot be applied. TreeID then proceeds to search missing cycles. One such cycle is $1 \leftrightarrow 2 \leftrightarrow 3 \leftrightarrow 4 \leftrightarrow 1$. Applying Theorem 1 to this cycle and simplifying the polynomials, yields a quadratic equation with

$$a = 0$$

$$\begin{aligned} b = &(\sigma_{01}\sigma_{02} - \sigma_{00}\sigma_{12})(\sigma_{04}\sigma_{33} - \sigma_{03}\sigma_{34}) \\ &- (\sigma_{03}\sigma_{14} - \sigma_{04}\sigma_{13})(\sigma_{00}\sigma_{23} - \sigma_{02}\sigma_{03}) \end{aligned}$$

$$\begin{aligned} c = &(\sigma_{01}\sigma_{02} - \sigma_{00}\sigma_{12})(\sigma_{13}\sigma_{34} - \sigma_{14}\sigma_{33}) \\ &- (\sigma_{03}\sigma_{14} - \sigma_{04}\sigma_{13})(\sigma_{03}\sigma_{12} - \sigma_{01}\sigma_{23}) \end{aligned}$$

Thus λ_1 is identified as $-c/b$. Every other effect λ_j of edge $q \to j$ can then be identified by propagation,

$\lambda_j = (\lambda_1\sigma_{0j} - \sigma_{1j})/(\lambda_1\sigma_{0q} - \sigma_{1q})$ using Corollary 2. Once the algorithm has found a solution for each edge, it is finished.

Since we did not specify an order of the cycles in the pseudocode of the algorithm, it might start with other cycles. If it first finds the missing cycle $1 \leftrightarrow 2 \leftrightarrow 3 \leftrightarrow 1$ and applies Theorem 1 there, it obtains a quadratic equation with $a \neq 0$ and two possible solutions for λ_1 involving a square root. It then has to continue searching cycles, and might find $1 \leftrightarrow 3 \leftrightarrow 4 \leftrightarrow 1$. Only one of the two previous solutions is also a solution for this cycle, so the algorithm eliminates one of them. The remaining solution for λ_1 then helps again to identify all other edges through propagation.

In $\mathcal{G}_3$, there are three missing cycles $1 \leftrightarrow 2 \leftrightarrow 3$, $2 \leftrightarrow 3 \leftrightarrow 4$, and $1 \leftrightarrow 2 \leftrightarrow 3 \leftrightarrow 4$. The missing cycle $1 \leftrightarrow 2 \leftrightarrow 3$ yields a quadratic equation with two solutions:

$\lambda_1 = (\sqrt{s} + (\sigma_{01}\sigma_{12} + \sigma_{02}\sigma_{11})\sigma_{23} + (-\sigma_{01}\sigma_{13} - \sigma_{03}\sigma_{11})\sigma_{22} - \sigma_{02}\sigma_{12}\sigma_{13} + \sigma_{03}\sigma_{12}^2)/(2\sigma_{01}\sigma_{02}\sigma_{23} - 2\sigma_{01}\sigma_{03}\sigma_{22} - 2\sigma_{02}^2\sigma_{13} + 2\sigma_{02}\sigma_{03}\sigma_{12})$, and

$\lambda_1' = (-\sqrt{s} + (\sigma_{01}\sigma_{12} + \sigma_{02}\sigma_{11})\sigma_{23} + (-\sigma_{01}\sigma_{13} - \sigma_{03}\sigma_{11})\sigma_{22} - \sigma_{02}\sigma_{12}\sigma_{13} + \sigma_{03}\sigma_{12}^2)/(2\sigma_{01}\sigma_{02}\sigma_{23} - 2\sigma_{01}\sigma_{03}\sigma_{22} - 2\sigma_{02}^2\sigma_{13} + 2\sigma_{02}\sigma_{03}\sigma_{12})$,

where $s = (\sigma_{01}^2\sigma_{12}^2 - 2\sigma_{01}\sigma_{02}\sigma_{11}\sigma_{12} + \sigma_{02}^2\sigma_{11}^2)\sigma_{23}^2 + (((2\sigma_{01}\sigma_{02}\sigma_{11} - 2\sigma_{01}^2\sigma_{12})\sigma_{13} + 2\sigma_{01}\sigma_{03}\sigma_{11}\sigma_{12} - 2\sigma_{02}\sigma_{03}\sigma_{11}^2)\sigma_{22} + (2\sigma_{02}^2\sigma_{11}\sigma_{12} - 2\sigma_{01}\sigma_{02}\sigma_{12}^2)\sigma_{13} + 2\sigma_{01}\sigma_{03}\sigma_{12}^3 - 2\sigma_{02}\sigma_{03}\sigma_{11}\sigma_{12}^2)\sigma_{23} + (\sigma_{01}^2\sigma_{13}^2 - 2\sigma_{01}\sigma_{03}\sigma_{11}\sigma_{13} + \sigma_{03}^2\sigma_{11}^2)\sigma_{22}^2 + ((2\sigma_{01}\sigma_{02}\sigma_{12} - 4\sigma_{02}^2\sigma_{11})\sigma_{13}^2 + (6\sigma_{02}\sigma_{03}\sigma_{11}\sigma_{12} - 2\sigma_{01}\sigma_{03}\sigma_{12}^2)\sigma_{13} - 2\sigma_{03}^2\sigma_{11}\sigma_{12}^2)\sigma_{22} + \sigma_{02}^2\sigma_{12}^2\sigma_{13}^2 - 2\sigma_{02}\sigma_{03}\sigma_{12}^3\sigma_{13} + \sigma_{03}^2\sigma_{12}^4$.

The solutions are distinct because $[\sqrt{s}]_{\mathcal{G}}$ simplifies to non-zero $\sqrt{(\lambda_{01}\omega_{01} + \omega_{11})^2(2\lambda_{01}\lambda_{12}\omega_{02} + \omega_{22})^2{\omega_{03}}^2}$.

Since there are two solutions, the algorithm continues searching cycles. It might find $1 \leftrightarrow 2 \leftrightarrow 3 \leftrightarrow 4$ next and conclude that the first solution λ_1 is the only solution. Knowing one solution, it can identify all other edges through propagation.

More details are given in the supplementary material.

7.2 Graphs unidentifiable by HTC and TSID

Weihs et al. (2018) have investigated the identifiability of all graphs with 5 nodes. There are 53 graphs in which each edge is uniquely identifiable using Gröbner bases, but that cannot be identified with the halftrek or TSID algorithm. Of these graphs, 15 are acyclic and 5 of those are trees. We show these trees in Fig. 3 and have applied our algorithm to them. There are bidirected edges from the root node to every other node, so the IV method cannot be used.

In the first graph, there are three missing cycles, $1 \leftrightarrow 2 \leftrightarrow 4$, $2 \leftrightarrow 3 \leftrightarrow 4$, and $1 \leftrightarrow 2 \leftrightarrow 3 \leftrightarrow 4$. Each cycle yields two solutions.

The solutions for λ_1 of $1 \leftrightarrow 2 \leftrightarrow 4$ are
$\lambda_1 = (\sqrt{s} + (\sigma_{01}\sigma_{13} + \sigma_{03}\sigma_{11})\sigma_{24} + (-\sigma_{01}\sigma_{14} - \sigma_{04}\sigma_{11})\sigma_{23} - \sigma_{03}\sigma_{12}\sigma_{14} + \sigma_{04}\sigma_{12}\sigma_{13})/(2\sigma_{01}\sigma_{03}\sigma_{24} - 2\sigma_{01}\sigma_{04}\sigma_{23} - 2\sigma_{02}\sigma_{03}\sigma_{14} + 2\sigma_{02}\sigma_{04}\sigma_{13})$ and
$\lambda_1' = (-\sqrt{s} + (\sigma_{01}\sigma_{13} + \sigma_{03}\sigma_{11})\sigma_{24} + (-\sigma_{01}\sigma_{14} - \sigma_{04}\sigma_{11})\sigma_{23} - \sigma_{03}\sigma_{12}\sigma_{14} + \sigma_{04}\sigma_{12}\sigma_{13})/(2\sigma_{01}\sigma_{03}\sigma_{24} - 2\sigma_{01}\sigma_{04}\sigma_{23} - 2\sigma_{02}\sigma_{03}\sigma_{14} + 2\sigma_{02}\sigma_{04}\sigma_{13})$
where $s = (-\sigma_{03}(\sigma_{12}\sigma_{14} - \sigma_{11}\sigma_{24}) + \sigma_{13}(\sigma_{01}\sigma_{24} - \sigma_{02}\sigma_{14}) - \sigma_{04}(\sigma_{11}\sigma_{23} - \sigma_{12}\sigma_{13}) + \sigma_{14}(\sigma_{02}\sigma_{13} - \sigma_{01}\sigma_{23}))^2 - 4(-\sigma_{03}(\sigma_{01}\sigma_{24} - \sigma_{02}\sigma_{14}) - \sigma_{04}(\sigma_{02}\sigma_{13} - \sigma_{01}\sigma_{23}))(\sigma_{13}(\sigma_{12}\sigma_{14} - \sigma_{11}\sigma_{24}) + \sigma_{14}(\sigma_{11}\sigma_{23} - \sigma_{12}\sigma_{13}))$.

The solutions are distinct because $[s]_{\mathcal{G}}$ simplifies to non-zero ${\omega_{04}}^2(\lambda_{01}\omega_{02}\omega_{13} + \lambda_{01}{\lambda_{12}}^2\lambda_{23}\omega_{01} - \lambda_{01}\lambda_{23}\omega_{01} + {\lambda_{01}}^2{\lambda_{12}}^2\lambda_{23} - {\lambda_{12}}^2\lambda_{23} - {\lambda_{01}}^2\lambda_{23} + \lambda_{23})^2$.

The former λ_1 is also a solution for the cycle $1 \leftrightarrow 2 \leftrightarrow 3 \leftrightarrow 4$. The latter λ_1' is not. Thus λ_1 is the true solution. Using propagate, all other edges are identifiable as well.

The same situation occurs in the other graphs. The individual cycles have two solutions, and the combination of a 3-cycle with a 4-cycle yields exactly one solution. The second graph is $\mathcal{G}_3$ in Figure 1. Further solutions are given in the supplementary material.

7.3 Path Graphs with a Single Missing Cycle

Now we consider path graphs, whose bidirected component is complete except for exactly one missing cycle. This is an important class to investigate to understand how many solutions Theorem 1 can yield.

First we show a way of transforming graphs into equivalent graphs. Since only the covariances between nodes with missing bidirected edges and their parents occur in the equations of Lemma 5, all other edges and nodes can be removed, added, or permutated without changing the solutions:

Lemma 8. *Let $\mathcal{G} = (V, D, B)$ be a path graph with nodes $0, \ldots, n$ and $0 \leftrightarrow i \in B$ for all $i \in \{1, \ldots, n\}$. Let $\mathbf{m} = \{i \mid \exists j (i \leftrightarrow j \notin B \vee i+1 \leftrightarrow j \notin B) \wedge i, j > 0\}$ be the nodes on directed edges into the missing cycle. The identifiability does not change if:*

- *Nodes not in $\mathbf{m} \cup 0$ are removed from the graph.*
- *The nodes are permutated by a permutation π with $\pi(0) = 0$; and, for all $i \in \mathbf{m}$ and $i+1 \in \mathbf{m}$, $\pi(i+1) = \pi(i) + 1$.*

Thus for each missing cycle path graph, there is a canonical graph, which can be obtained by permutating all nodes not affecting the missing cycle to the

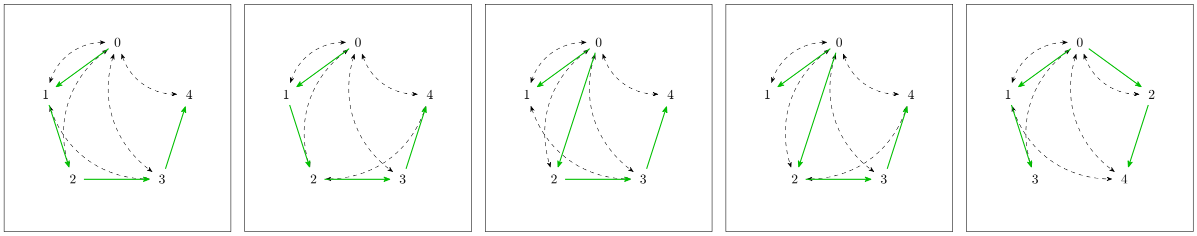


Figure 3: Tree graphs of (Weihs et al., 2018), where each directed edge is identifiable (green). The authors name them (4680, 403), (4680, 914), (360, 117), (360, 369), (840, 466). We relabel the nodes to make node 0 the root.

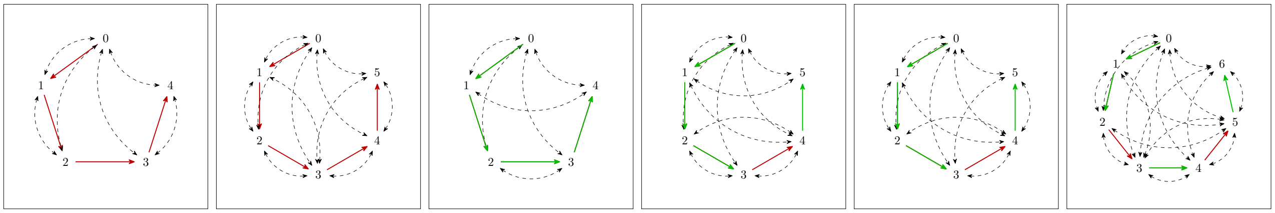


Figure 4: The two unidentifiable and the four uniquely identifiable path graphs with a single missing cycle. Identifiable edges are shown in green and unidentifiable edges in red.

end of the path and removing them. For a certain cycle length, there are only finitely many canonical graphs. Enumerating all canonical graphs up to length 10, shows there are only 6 missing cycles whose incoming edges are not 2-identifiable (see Figure 4). The first two are not identifiable at all, the other four have a unique solution for all edges into the missing cycle:

- $1 \leftrightarrow 3 \leftrightarrow 2 \leftrightarrow 4 \leftrightarrow 1$ (unidentifiable)
- $1 \leftrightarrow 4 \leftrightarrow 2 \leftrightarrow 5 \leftrightarrow 1$ (unidentifiable)
- $1 \leftrightarrow 2 \leftrightarrow 4 \leftrightarrow 3 \leftrightarrow 1$ (uniquely identifiable)
- $1 \leftrightarrow 2 \leftrightarrow 5 \leftrightarrow 3 \leftrightarrow 1$ (uniquely identifiable)
- $1 \leftrightarrow 3 \leftrightarrow 2 \leftrightarrow 5 \leftrightarrow 1$ (uniquely identifiable)
- $1 \leftrightarrow 4 \leftrightarrow 2 \leftrightarrow 6 \leftrightarrow 1$ (uniquely identifiable)

Proposition 5. *Any path graph with exactly one missing cycle that is not equivalent (after transformations with Lemma 8) to one of the graphs of Fig. 4 is 2-identifiable, if the cycle length is at most 10.*

The above results have been obtained using Gröbner bases as a reference solution. This enumeration was only possible because the Gröbner base calculation performed vastly faster on path graphs with a single missing cycle than on path graphs with arbitrarily missing bidirected edges.

Our algorithm can be applied to the graphs in Figure 4 and Theorem 1 returns quadratic equations for the missing cycles. For the unidentifiable graphs, the quadratic equation vanishes that is $[a]_{\mathcal{G}} = [b+c]_{\mathcal{G}} = [d]_{\mathcal{G}} = 0$. For the identifiable graphs, there is one edge for which the theorem returns a linear equation, i.e. $[a]_{\mathcal{G}} = 0$, which implies that all incoming edges are identifiable with propagation. The exact results are given in the supplementary material.

8 CONCLUSIONS

Our algorithm allows the identification of causal effects in tree graphs which could not be identified previously without Gröbner bases. It is possible that the algorithm is complete for the considered family of causal models, i.e., it is able to identify all causal effects in tree graphs if and only if the effects are identifiable by any method, although we cannot prove that. However, to our knowledge, no algorithm is known so far that is complete for a certain natural family of graphs.

In the worst case, the algorithm enumerates all missing cycles, although that is unnecessary since two cycles for each edge are always sufficient to identify an identifiable edge. An open problem remains to find just those two cycles efficiently. The next issue is that the solutions provided by the algorithm are in the form of fairly complex expressions, so their numerical stability on real life datasets needs further investigation.

Future research might generalize the algorithm to further graph classes. The missing cycle method of Theorem 1 only requires that all nodes on the missing cycle have one and only one incoming directed edge. If the resulting equation systems satisfy the conditions of Lemma 2, our algorithm can probably already be used to identify the incoming edges in more complex graphs that only contain a tree as subgraph.

We have implemented our algorithm for the DAGitty project (`www.dagitty.net`, see Textor et al. (2016)). To ensure its correctness, we have compared this implementation with a Gröbner base implementation (see supplementary material B.4).

Acknowledgments The first author gratefully acknowledges the financial support by the Federal Ministry for Economic Affairs and Climate Action of Germany (BMWK) through the KI-SIGS Project (FKZ: 01MK20012B). This work was also supported by the Deutsche Forschungsgemeinschaft (DFG) grant 471183316 (ZA 1244/1-1).

References

M. Bläser, D. Eisenbud, and F.-O. Schreyer. Ulrich complexity. *Differential Geometry and its Applications*, 55:128–145, 2017.

K. A. Bollen. *Structural equations with latent variables.* John Wiley & Sons, 1989.

R. Bowden and D. Turkington. *Instrumental variables.* Cambridge University Press, 1984.

R. P. Brent and H. T. Kung. Fast algorithms for manipulating formal power series. *Journal of the ACM*, 25(4):581–595, 1978.

C. Brito. Instrumental sets. In R. Dechter, H. Geffner, and J. Y. Halpern, editors, *Heuristics, Probability and Causality. A Tribute to Judea Pearl*, chapter 17, pages 295–308. College Publications, 2010.

C. Brito and J. Pearl. Generalized instrumental variables. In *Proc. Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 85–93, 2002a.

C. Brito and J. Pearl. A graphical criterion for the identification of causal effects in linear models. In *Proc. AAAI Conference on Artificial Intelligence*, pages 533–538, 2002b.

B. Chen, J. Pearl, and E. Bareinboim. Incorporating knowledge into structural equation models using auxiliary variables. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3577–3583, 2015.

B. Chen, D. Kumor, and E. Bareinboim. Identification and model testing in linear structural equation models using auxiliary variables. In *Proc. International Conference on Machine Learning (ICML)*, pages 757–766. PMLR, 2017.

W. Decker, G.-M. Greuel, G. Pfister, and H. Schönemann. Singular 4-0-3 — A computer algebra system for polynomial computations. `http://www.singular.uni-kl.de`, 2016.

R. A. DeMillo and R. J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193–195, 1978.

M. Drton. Algebraic problems in structural equation modeling. In *The 50th anniversary of Gröbner bases*, pages 35–86. Mathematical Society of Japan, 2018.

O. D. Duncan. *Introduction to structural equation models.* Academic Press, 1975.

F. M. Fisher. *The identification problem in econometrics.* McGraw-Hill, 1966.

R. Foygel, J. Draisma, and M. Drton. Half-trek criterion for generic identifiability of linear structural equation models. *The Annals of Statistics*, 40(3): 1682–1713, 06 2012.

L. D. García-Puente, S. Spielvogel, and S. Sullivant. Identifying Causal Effects with Computer Algebra. In *Proc. Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 193–200. AUAI Press, 2010.

D. Kumor, B. Chen, and E. Bareinboim. Efficient identification in linear structural causal models with instrumental cutsets. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pages 12477–12486, 2019.

D. Kumor, C. Cinelli, and E. Bareinboim. Efficient identification in linear structural causal models with auxiliary cutsets. In *Proc. International Conference on Machine Learning (ICML)*, pages 5501–5510. PMLR, 2020.

E. W. Mayr and A. R. Meyer. The complexity of the word problems for commutative semigroups and polynomial ideals. *Advances in Mathematics*, 46(3): 305–329, 1982.

J. Pearl. Parameter identification: A new perspective. Technical Report R-276, UCLA, 2001.

J. Pearl. *Causality.* Cambridge University Press, 2009. ISBN 0-521-77362-8.

J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, 1980.

J. Textor, B. van der Zander, M. S. Gilthorpe, M. Liśkiewicz, and G. T. Ellison. Robust causal inference using directed acyclic graphs: the R package 'dagitty'. *International Journal of Epidemiology*, 45 (6):1887–1894, 2016.

B. van der Zander and M. Liśkiewicz. On searching for generalized instrumental variables. In *Proc. International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1214–1222, 2016.

B. van der Zander, J. Textor, and M. Liśkiewicz. Efficiently finding conditional instruments for causal inference. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3243–3249, 2015.

L. Weihs, B. Robinson, E. Dufresne, J. Kenkel, K. K. R. McGee II, M. I. Reginald, N. Nguyen, E. Robeva, and M. Drton. Determinantal generalizations of instrumental variables. *Journal of Causal Inference*, 6(1), 2018.

P. G. Wright. *Tariff on animal and vegetable oils.* Macmillan Company, New York, 1928.

S. Wright. Correlation and causation. *J. Agricultural Research*, 20:557–585, 1921.

S. Wright. The method of path coefficients. *The Annals of Mathematical Statistics*, 5(3):161–215, 1934.

R. Zippel. Probabilistic algorithms for sparse polynomials. *Symbolic and algebraic computation*, pages 216–226, 1979.

Supplementary Material:
Identification in Tree-shaped Linear Structural Causal Models

A MISSING PROOFS

A.1 Proof of Lemma 3

Proof. According to the trek rule (2) we know that σ_{ij} is given by the sum of the products along a trek over all treks between i and j. A trek containing the bidirected edge $s \leftrightarrow t$ contributes $\omega_{st} L(s,i) L(t,j)$. A trek without a bidirected edge contributes $\omega_{ss} L(s,i) L(s,j)$, for $s = t \in An(i) \cap An(j)$. □

A.2 Proof of Lemma 4

Proof. For an arbitrary mixed graph and the corresponding matrix $\Omega = (I - \Lambda)^T \Sigma (I - \Lambda)$, Drton shows (2018) that $[(I-\Lambda)^T \Sigma (I-\Lambda)]_{ij} = \sum_{p \in Pa(i)} \sum_{q \in Pa(j)} \lambda_{pi} \lambda_{qj} \sigma_{pq} - \sum_{p \in Pa(i)} \lambda_{pi} \sigma_{pj} - \sum_{q \in Pa(j)} \lambda_{qj} \sigma_{pi} + \sigma_{ij}$. Our lemma for tree graphs follows from the equation. □

An alternate proof can be given directly using Wright's path rules.

Proof. We show first that $[\lambda_{pi} \lambda_{qj} \sigma_{pq} - \lambda_{pi} \sigma_{pj} - \lambda_{qj} \sigma_{iq} + \sigma_{ij}]_{\mathcal{G}} = \omega_{ij}$. Indeed,

$$[\lambda_{p,i} \lambda_{q,j} \sigma_{p,q} - \lambda_{p,i} \sigma_{p,j} - \lambda_{q,j} \sigma_{i,q} + \sigma_{i,j}]_{\mathcal{G}}$$

$=$

$$\begin{aligned} \lambda_{p,i} \lambda_{q,j} & \left(\sum_{s \in An(p)} \sum_{t \in An(q)} \omega_{s,t} L(s,p) L(t,q) \right) \\ -\lambda_{p,i} & \left(\sum_{s \in An(p)} \sum_{t \in An(j)} \omega_{s,t} L(s,p) L(t,j) \right) \\ -\lambda_{q,j} & \left(\sum_{s \in An(i)} \sum_{t \in An(q)} \omega_{s,t} L(s,i) L(t,q) \right) \\ & + \sum_{s \in An(i)} \sum_{t \in An(j)} \omega_{s,t} L(s,i) L(t,j) \end{aligned}$$

$=$

$$\begin{aligned}
&\left(\sum_{s\in An(p)}\sum_{t\in An(q)}\omega_{s,t}L(s,p)\lambda_{p,i}L(t,q)\lambda_{q,j}\right)\\
&-\left(\sum_{s\in An(p)}\sum_{t\in An(j)}\omega_{s,t}L(s,p)\lambda_{p,i}L(t,j)\right)\\
&-\left(\sum_{s\in An(i)}\sum_{t\in An(q)}\omega_{s,t}L(s,i)L(t,q)\lambda_{q,j}\right)\\
&+\sum_{s\in An(i)}\sum_{t\in An(j)}\omega_{s,t}L(s,i)L(t,j)
\end{aligned}$$

$$=$$

$$\begin{aligned}
&\sum_{s\in An(p)}\sum_{t\in An(q)}\omega_{s,t}L(s,i)L(t,j)\\
&-\sum_{s\in An(p)}\sum_{t\in An(j)}\omega_{s,t}L(s,i)L(t,j)\\
&-\sum_{s\in An(i)}\sum_{t\in An(q)}\omega_{s,t}L(s,i)L(t,j)\\
&+\sum_{s\in An(i)}\sum_{t\in An(j)}\omega_{s,t}L(s,i)L(t,j)
\end{aligned}$$

$$=$$

$$\begin{aligned}
&\sum_{s\in An(p)}\sum_{t\in An(q)}\omega_{s,t}L(s,i)L(t,j)\\
&-\sum_{s\in An(p)}\sum_{t\in An(q)\cup j}\omega_{s,t}L(s,i)L(t,j)\\
&-\sum_{s\in An(p)\cup i}\sum_{t\in An(q)}\omega_{s,t}L(s,i)L(t,j)\\
&+\sum_{s\in An(p)\cup i}\sum_{t\in An(q)\cup j}\omega_{s,t}L(s,i)L(t,j)
\end{aligned}$$

$$=$$

$$\begin{aligned}
&-\sum_{s\in An(p)}\omega_{s,j}L(s,i)L(j,j)\\
&+\sum_{s\in An(p)\cup i}\omega_{s,j}L(s,i)L(j,j)
\end{aligned}$$

$= \omega_{i,j} L(i,i) L(j,j) = \omega_{i,j}.$

Now, we show that $[\sigma_{0i} - \lambda_{qi}\sigma_{0q}]_{\mathcal{G}} = \omega_{0i}$:

$$[\sigma_{0,i} - \lambda_{p,i}\sigma_{0,p}]_{\mathcal{G}}$$

$=$

$$\sum_{s \in An(0)} \sum_{t \in An(i)} \omega_{s,t} L(s,0) L(t,i)$$
$$-\lambda_{p,i} \left(\sum_{s \in An(0)} \sum_{t \in An(p)} \omega_{s,t} L(s,0) L(t,p) \right)$$

$=$

$$\sum_{s \in An(0)} \sum_{t \in An(i)} \omega_{s,t} L(s,0) L(t,i)$$
$$- \left(\sum_{s \in An(0)} \sum_{t \in An(p)} \omega_{s,t} L(s,0) L(t,p) \lambda_{p,i} \right)$$

$=$

$$\sum_{s \in An(0)} \sum_{t \in An(i)} \omega_{s,t} L(s,0) L(t,i)$$
$$- \sum_{s \in An(0)} \sum_{t \in An(p)} \omega_{s,t} L(s,0) L(t,i)$$

$=$

$$\sum_{t \in An(p) \cup i} \omega_{0,t} L(0,0) L(t,i)$$
$$- \sum_{t \in An(p)} \omega_{0,t} L(0,0) L(t,i)$$

$= \omega_{0,i} L(0,0) L(i,i) = \omega_{0,i}.$ □

A.3 Proof of Lemma 5

Proof. If edge $i \leftrightarrow j$ is missing, then $\omega_{ij} = 0$. So from Lemma 4, it follows immediately, that the degree of identifiability is at most the number of generic solutions of this equation system.

Foygel et al. (2012) have proven that the degree of identifiability is given exactly by number of solutions of the system $(I - \Lambda)\Sigma(I - \Lambda)^T = \Omega$ on the elements corresponding to missing bidirected edges. If one calculates the matrix $(I - \Lambda)\Sigma(I - \Lambda)^T$, one obtains this equation system. □

A.4 Proof of Proposition 1

Proof. In the preliminary identification, the root is used as instrumental variable (Corollary 1) for identifying the edge $x \to y$, in case there is no edge $0 \leftrightarrow y$. It can be easily seen that, if this criterion does not apply, there can be no instrument z for $x \to y$ as there is an open backdoor path from z via the root to y. Hence, using Corollary 1 is equivalent to IV.

Whenever an edge is identified, aIV creates an auxiliary variable to use as instrument. Propagation (Lemma 6) applies the same criterion by ensuring that there is a path from instrument z via x to y in the graph with $\to z$ removed and no backdoor path (which again holds iff there is no bidirected edge between z and y). □

A.5 Proof of Proposition 2

Proof. ACID repeatedly identifies direct effects (and partial effects) and creates corresponding auxiliary variables. For this, two criteria are used, which both collapse to the IV setting on tree graphs as we show below. Hence, in each iteration, the same direct effects are identified and the same auxiliary variables created as in aIV. It follows that the whole method is equivalent to aIV on tree graphs.

We now consider the two criteria for the identification of vertex y with parent x.

1. The instrumental cutset (IC) criterion (originally proposed by (Kumor et al., 2019, Theorem 5.1)):

   As each node has in-degree 1, we have that $|S| = 1$ and $|T| = 0$. Any $s \in S$ fulfilling the three IC criteria is an instrument for $x \to y$.

2. The auxiliary cutset (AC) criterion (Kumor et al., 2020, Definition 3.2): The auxiliary cutset is defined as the closest cutset to $x = Pa(y)$. In graphs with in-degree 1, it is always x itself. Then, a set Z which acts as partial-effect instrumental set (PEIS), see (Kumor et al., 2020, Definition 3.1), has cardinality 1 and $z \in Z$ is an instrumental variable.

This shows that, restricted to tree graphs, every edge identified by ACID is identified by aIV. Thus from Proposition 1 the claim of this proposition follows. □

A.6 Proof of Theorem 1

Proof. Due to Lemma 5, the missing cycle yields k equations, given by $a_i^1 \lambda_{v_i} \lambda_{v_{i+1}} + b_i^1 \lambda_{v_i} + c_i^1 \lambda_{v_{i+1}} + d_i^1 = 0$.

We can eliminate every other equation by combining pairs of equations. This is a general approach which works for all equation systems of this structure even if they do not come from a missing cycle.

Let, for short, $x_i = \lambda_{v_i}$ and $x_{k+1} = x_1$. We combine the $(2i-1)$th with the $(2i)$th equation:

$$\begin{aligned}
0 &= (x_{2i-1}a_{2i-1} + c_{2i-1})(a_{2i}x_{2i}x_{2i+1} + b_{2i}x_{2i} + c_{2i}x_{2i+1} + d_{2i}) \\
&\quad - (x_{2i+1}a_{2i} + b_{2i})(a_{2i-1}x_{2i-1}x_{2i} + b_{2i-1}x_{2i-1} + c_{2i-1}x_{2i} + d_{2i-1}) \\
&= (x_{2i-1}a_{2i-1} + c_{2i-1})(x_{2i}(a_{2i}x_{2i+1} + b_{2i}) + c_{2i}x_{2i+1} + d_{2i}) \\
&\quad - (x_{2i+1}a_{2i} + b_{2i})(x_{2i}(a_{2i-1}x_{2i-1} + c_{2i-1}) + b_{2i-1}x_{2i-1} + d_{2i-1}) \\
&= (x_{2i-1}a_{2i-1} + c_{2i-1})(c_{2i}x_{2i+1} + d_{2i}) \\
&\quad - (x_{2i+1}a_{2i} + b_{2i})(b_{2i-1}x_{2i-1} + d_{2i-1}) \\
&= x_{2i-1}x_{2i+1}(a_{2i-1}c_{2i} - a_{2i}b_{2i-1}) \\
&\quad + x_{2i-1}(a_{2i-1}d_{2i} - b_{2i-1}b_{2i}) \\
&\quad + x_{2i+1}(c_{2i-1}c_{2i} - a_{2i}d_{2i-1}) \\
&\quad + c_{2i-1}d_{2i} - d_{2i-1}b_{2i}.
\end{aligned}$$

This eliminates every equation involving x_{2i}. If k is even, it results in a new equation system of size $k/2$. If k is odd, we can do the same, but include the last equation in the new equation system. Any solution of the old system is a solution of the new system.

Once two equations, as if $k = 2$, remain, we eliminate the second variable by:

$$\begin{aligned}0 &= (a_1x_1 + c_1)(a_2x_2x_1 + b_2x_2 + c_2x_1 + d_2)\\ &\quad - (a_2x_1 + b_2)(a_1x_1x_2 + b_1x_1 + c_1x_2 + d_1)\\ &= (a_1x_1 + c_1)(x_2(a_2x_1 + b_2) + c_2x_1 + d_2)\\ &\quad - (a_2x_1 + b_2)(x_2(a_1x_1 + c_1) + b_1x_1 + d_1)\\ &= (a_1x_1 + c_1)(c_2x_1 + d_2) - (a_2x_1 + b_2)(b_1x_1 + d_1)\\ &= x_1^2(a_1c_2 - a_2b_1) + x_1(a_1d_2 + c_1c_2 - a_2d_1 - b_2b_1) + c_1d_2 - b_2d_1.\end{aligned}$$

Finally, only one quadratic equation remains.

The coefficients of the resulting equations are exactly the determinants calculated in Definition 1. □

The proof shows that any solution to the initial equation system of one missing cycle is a solution to the final quadratic equation of the recursion. Since a quadratic equation has at most two solutions, this means if the equation system has at least two solutions, the solutions of the quadratic equation are exactly the solutions of the equation system.

Unfortunately, if the equation system has exactly one solution, the recursion can introduce a spurious second solution. E.g. if it happens that $a_1 = a_2$ and $c_1 = b_2$ in the two equations case. Then $x_1 = -c_1/a_1 = -b_2/a_2$ is a solution to the final quadratic equation, regardless if it was a solution of the initial equation system.

This unfortunate case actually happened for some edges in Figure 4 during our experiments. However, it does not affect the outcome of the algorithm since there always was an edge in the same cycle for which it did not happen. This edge provides a unique solution for all other edges in the cycle using propagate.

An alternate way of solving the missing cycle equation system is to insert the equation of propagate into the next equation. E.g. from $a_1x_1x_2 + b_1x_1 + c_1x_2 + d_1 = 0$ obtain $x_2 = -(d_1 + b_1x_1)/(a_1x_1 + c_1)$ and insert it into $a_2x_2x_1+b_2x_2+c_2x_1+d_2$. This returns $a_2(-(d_1+b_1x_1)/(a_1x_1+c_1))x_1+b_2(-(d_1+b_1x_1)/(a_1x_1+c_1))+c_2x_1+d_2 = 0$, which can be solved for x_1. But we do not recommend that approach, since it requires linearly many insertion steps in general unlike the recursion we propose which only has logarithmic deep, so it has worse complexity and yields much larger expressions.

A.7 Proof of Proposition 4

Proof. The theorem follows from Proposition 2 which says that every edge in a tree graph identified by the ACID algorithm is identified already in the preliminary identification phase of the TreeID algorithm (Lines 24-27). We note that the phase of the algorithm runs in polynomial time. □

A.8 Proof of Lemma 8

Proof. The first part follows directly because nodes connected to all other nodes by bidirected edges do not contribute an equation to the equation system.

The second part: The equation of a missing bidirected edge $i + 1 \leftrightarrow j + 1$ contains factors $\sigma_{i+x,j+y}$ with $x, y \in \{0, 1\}$ in $\mathcal{G}$ and $\sigma_{\pi(i)+x,\pi(j)+y}$ in $\mathcal{G}'$. Then $i, j \in \mathbf{m}$ and $\pi(i) + x = \pi(i + x)$ and $\pi(j) + y = \pi(j + y)$.

So the new equation system is obtained by replacing $\sigma_{i,j}$ with $\sigma_{\pi(i),\pi(j)}$, λ_i with $\lambda_{\pi(i)}$, which is only a renaming of variables. Thus both equation systems have an isomorph solution space. □

A.9 Proof of Proposition 5

Proof. See Subsection B.3 below. □

B EXPERIMENTS

To test our algorithm TreeId on graphs, we have manually searched the missing cycles, and solved the resulting missing cycle equations. If multiple solutions occurred, we have inserted the solution of one missing cycle in the equations of other missing cycles.

For symbolic calculations with polynomials, we have used the CAS (wx)Maxima. Rather than using PIT on FASTPs, we have fully expanded the equations in the CAS. Although this is slower, it provides more detailed information about the terms remaining in non-zero polynomials. A problem that occurs during the calculations is that Maxima converts $\sqrt{x^2}$ to $|x|$, so that $\sqrt{x^2} - x$ is non-zero. For TreeId to work, such equations should be considered as zero and need to be manually checked.

We only need to calculate one edge, since the identifiability of other edges should follow using propagation.

B.1 Identification of $\mathcal{G}_2$ in Figure 1

The graph $\mathcal{G}_2$ in Figure 1 has a missing cycle $1 \leftrightarrow 2 \leftrightarrow 3 \leftrightarrow 4 \leftrightarrow 1$.

The recursion returns a quadratic equation $a\lambda_{01}^2 + b\lambda_{01} + c = 0$ with

$a = ((\sigma_{00}(-\sigma_{02}) - \sigma_{00}(-\sigma_{02}))((-\sigma_{33})(-\sigma_{04}) - \sigma_{03}\sigma_{34}) - (\sigma_{03}(-\sigma_{04}) - \sigma_{03}(-\sigma_{04}))(\sigma_{00}\sigma_{23} - (-\sigma_{03})(-\sigma_{02})))$

$b = ((\sigma_{00}(-\sigma_{02}) - \sigma_{00}(-\sigma_{02}))((-\sigma_{33})\sigma_{14} - (-\sigma_{13})\sigma_{34}) - (\sigma_{03}\sigma_{14} - (-\sigma_{13})(-\sigma_{04}))(\sigma_{00}\sigma_{23} - (-\sigma_{03})(-\sigma_{02}))) + (((-\sigma_{01})(-\sigma_{02}) - \sigma_{00}\sigma_{12})((-\sigma_{33})(-\sigma_{04}) - \sigma_{03}\sigma_{34}) - (\sigma_{03}(-\sigma_{04}) - \sigma_{03}(-\sigma_{04}))((-\sigma_{01})\sigma_{23} - (-\sigma_{03})\sigma_{12}))$

$c = (((-\sigma_{01})(-\sigma_{02}) - \sigma_{00}\sigma_{12})((-\sigma_{33})\sigma_{14} - (-\sigma_{13})\sigma_{34}) - (\sigma_{03}\sigma_{14} - (-\sigma_{13})(-\sigma_{04}))((-\sigma_{01})\sigma_{23} - (-\sigma_{03})\sigma_{12}))$

Simplifying this equation in (wx)Maxima returns:

$a = 0$

$b = (\sigma_{01}\sigma_{02} - \sigma_{00}\sigma_{12})(\sigma_{04}\sigma_{33} - \sigma_{03}\sigma_{34}) - (\sigma_{03}\sigma_{14} - \sigma_{04}\sigma_{13})(\sigma_{00}\sigma_{23} - \sigma_{02}\sigma_{03})$

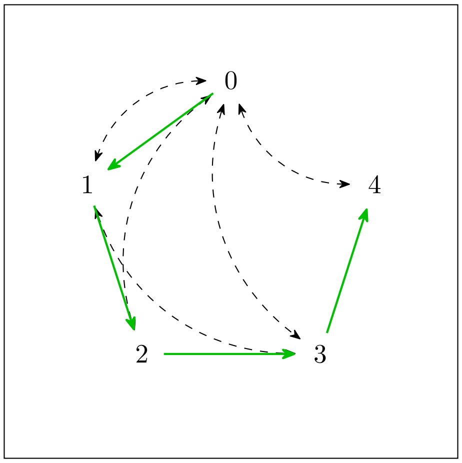
$c = (\sigma_{01}\sigma_{02} - \sigma_{00}\sigma_{12})(\sigma_{13}\sigma_{34} - \sigma_{14}\sigma_{33}) - (\sigma_{03}\sigma_{14} - \sigma_{04}\sigma_{13})(\sigma_{03}\sigma_{12} - \sigma_{01}\sigma_{23})$

Thus $\lambda_{01} = -c/b$ is a unique solution. The fraction is valid because $[b]_{\mathcal{G}} = \omega_{33}\omega_{01}\omega_{02}\omega_{04}$ is not zero.

B.2 Identification of the graphs in Figure 3

Here we investigate the 5 tree graphs of (Weihs et al., 2018).

B.2.1 Identification of (4680, 403)



There are three missing cycles, $1 \leftrightarrow 2 \leftrightarrow 4$, $2 \leftrightarrow 3 \leftrightarrow 4$, and $1 \leftrightarrow 2 \leftrightarrow 3 \leftrightarrow 4$. Each cycle yields two solutions. The solutions for λ_1 of $1 \leftrightarrow 2 \leftrightarrow 4$ are

$\lambda_1 = (\sqrt{s} + (\sigma_{01}\sigma_{13} + \sigma_{03}\sigma_{11})\sigma_{24} + (-\sigma_{01}\sigma_{14} - \sigma_{04}\sigma_{11})\sigma_{23} - \sigma_{03}\sigma_{12}\sigma_{14} + \sigma_{04}\sigma_{12}\sigma_{13})/(2\sigma_{01}\sigma_{03}\sigma_{24} - 2\sigma_{01}\sigma_{04}\sigma_{23} - 2\sigma_{02}\sigma_{03}\sigma_{14} + 2\sigma_{02}\sigma_{04}\sigma_{13})$, and

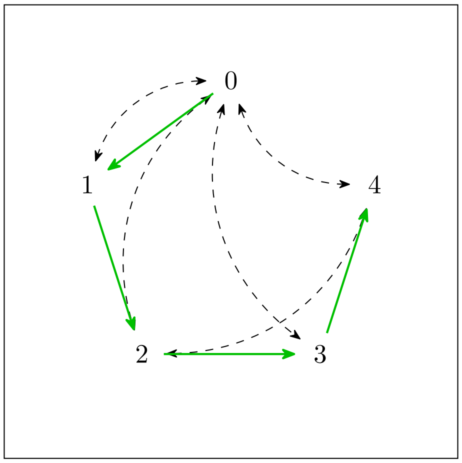
$\lambda_1' = (-\sqrt{s} + (\sigma_{01}\sigma_{13} + \sigma_{03}\sigma_{11})\sigma_{24} + (-\sigma_{01}\sigma_{14} - \sigma_{04}\sigma_{11})\sigma_{23} - \sigma_{03}\sigma_{12}\sigma_{14} + \sigma_{04}\sigma_{12}\sigma_{13})/(2\sigma_{01}\sigma_{03}\sigma_{24} - 2\sigma_{01}\sigma_{04}\sigma_{23} - 2\sigma_{02}\sigma_{03}\sigma_{14} + 2\sigma_{02}\sigma_{04}\sigma_{13})$

where $s = (-\sigma_{03}(\sigma_{12}\sigma_{14} - \sigma_{11}\sigma_{24}) + \sigma_{13}(\sigma_{01}\sigma_{24} - \sigma_{02}\sigma_{14}) - \sigma_{04}(\sigma_{11}\sigma_{23} - \sigma_{12}\sigma_{13}) + \sigma_{14}(\sigma_{02}\sigma_{13} - \sigma_{01}\sigma_{23}))^2 - 4(-\sigma_{03}(\sigma_{01}\sigma_{24} - \sigma_{02}\sigma_{14}) - \sigma_{04}(\sigma_{02}\sigma_{13} - \sigma_{01}\sigma_{23}))(\sigma_{13}(\sigma_{12}\sigma_{14} - \sigma_{11}\sigma_{24}) + \sigma_{14}(\sigma_{11}\sigma_{23} - \sigma_{12}\sigma_{13}))$.

The solutions are distinct because the expression $[\sqrt{s}]_{\mathcal{G}}$ simplifies to the non-zero polynomial $\sqrt{{\omega_{04}}^2(\lambda_{01}\omega_{02}\omega_{13} + \lambda_{01}{\lambda_{12}}^2\lambda_{23}\omega_{01} - \lambda_{01}\lambda_{23}\omega_{01} + {\lambda_{01}}^2{\lambda_{12}}^2\lambda_{23} - {\lambda_{12}}^2\lambda_{23} - {\lambda_{01}}^2\lambda_{23} + \lambda_{23})^2}$.

The former λ_1 is also a solution for the cycle $1 \leftrightarrow 2 \leftrightarrow 3 \leftrightarrow 4$. The latter λ_1' is not. Thus λ_1 is the true solution.

B.2.2 Identification of (4680, 914)



There are three missing cycles $1 \leftrightarrow 2 \leftrightarrow 3$, $2 \leftrightarrow 3 \leftrightarrow 4$, and $1 \leftrightarrow 2 \leftrightarrow 3 \leftrightarrow 4$. The missing cycle $1 \leftrightarrow 2 \leftrightarrow 3$ gives two solutions:

$\lambda_1 = (\sqrt{s} + (\sigma_{01}\sigma_{12} + \sigma_{02}\sigma_{11})\sigma_{23} + (-\sigma_{01}\sigma_{13} - \sigma_{03}\sigma_{11})\sigma_{22} - \sigma_{02}\sigma_{12}\sigma_{13} + \sigma_{03}\sigma_{12}^2)/(2\sigma_{01}\sigma_{02}\sigma_{23} - 2\sigma_{01}\sigma_{03}\sigma_{22} - 2\sigma_{02}^2\sigma_{13} + 2\sigma_{02}\sigma_{03}\sigma_{12})$, and

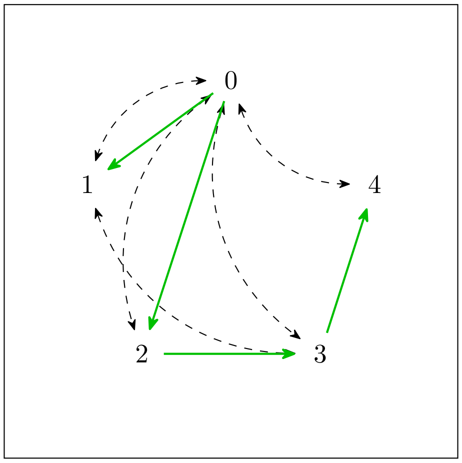
$\lambda_1' = (-\sqrt{s} + (\sigma_{01}\sigma_{12} + \sigma_{02}\sigma_{11})\sigma_{23} + (-\sigma_{01}\sigma_{13} - \sigma_{03}\sigma_{11})\sigma_{22} - \sigma_{02}\sigma_{12}\sigma_{13} + \sigma_{03}\sigma_{12}^2)/(2\sigma_{01}\sigma_{02}\sigma_{23} - 2\sigma_{01}\sigma_{03}\sigma_{22} - 2\sigma_{02}^2\sigma_{13} + 2\sigma_{02}\sigma_{03}\sigma_{12})$

where $s = (\sigma_{01}^2\sigma_{12}^2 - 2\sigma_{01}\sigma_{02}\sigma_{11}\sigma_{12} + \sigma_{02}^2\sigma_{11}^2)\sigma_{23}^2 + (((2\sigma_{01}\sigma_{02}\sigma_{11} - 2\sigma_{01}^2\sigma_{12})\sigma_{13} + 2\sigma_{01}\sigma_{03}\sigma_{11}\sigma_{12} - 2\sigma_{02}\sigma_{03}\sigma_{11}^2)\sigma_{22} + (2\sigma_{02}^2\sigma_{11}\sigma_{12} - 2\sigma_{01}\sigma_{02}\sigma_{12}^2)\sigma_{13} + 2\sigma_{01}\sigma_{03}\sigma_{12}^3 - 2\sigma_{02}\sigma_{03}\sigma_{11}\sigma_{12}^2)\sigma_{23} + (\sigma_{01}^2\sigma_{13}^2 - 2\sigma_{01}\sigma_{03}\sigma_{11}\sigma_{13} + \sigma_{03}^2\sigma_{11}^2)\sigma_{22}^2 + ((2\sigma_{01}\sigma_{02}\sigma_{12} - 4\sigma_{02}^2\sigma_{11})\sigma_{13}^2 + (6\sigma_{02}\sigma_{03}\sigma_{11}\sigma_{12} - 2\sigma_{01}\sigma_{03}\sigma_{12}^2)\sigma_{13} - 2\sigma_{03}^2\sigma_{11}\sigma_{12}^2)\sigma_{22} + \sigma_{02}^2\sigma_{12}^2\sigma_{13}^2 - 2\sigma_{02}\sigma_{03}\sigma_{12}^3\sigma_{13} + \sigma_{03}^2\sigma_{12}^4$.

The solutions are distinct because $[\sqrt{s}]_{\mathcal{G}}$ simplifies to non-zero $\sqrt{{(\lambda_{01}\omega_{01} + \omega_{11})}^2{(2\lambda_{01}\lambda_{12}\omega_{02} + \omega_{22})}^2{\omega_{03}}^2}$.

If we insert this in the cycle $1 \leftrightarrow 2 \leftrightarrow 3 \leftrightarrow 4$, maxima says neither is a solution because there are terms involving $\mid w_{03} \mid$ that do not cancel. Manually replacing them with (w_{03}) discovers that only the first λ_1 is a solution. Thus the graph is fully identifiable.

B.2.3 Identification of (360, 117)



The missing cycle $1 \leftrightarrow 2 \leftrightarrow 4$ gives two solutions:

$\lambda_1 = (\sqrt{s} + (\sigma_{00}\sigma_{13} + \sigma_{01}\sigma_{03})\sigma_{24} + (-\sigma_{00}\sigma_{14} - \sigma_{01}\sigma_{04})\sigma_{23} + \sigma_{02}\sigma_{03}\sigma_{14} - \sigma_{02}\sigma_{04}\sigma_{13})/(2\sigma_{00}\sigma_{03}\sigma_{24} - 2\sigma_{00}\sigma_{04}\sigma_{23})$, and

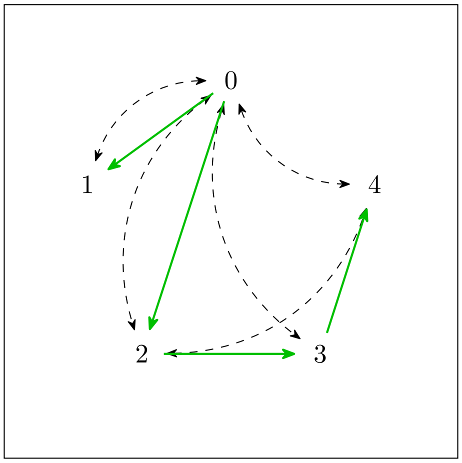
$\lambda_1' = (-\sqrt{s} + (\sigma_{00}\sigma_{13} + \sigma_{01}\sigma_{03})\sigma_{24} + (-\sigma_{00}\sigma_{14} - \sigma_{01}\sigma_{04})\sigma_{23} + \sigma_{02}\sigma_{03}\sigma_{14} - \sigma_{02}\sigma_{04}\sigma_{13})/(2\sigma_{00}\sigma_{03}\sigma_{24} - 2\sigma_{00}\sigma_{04}\sigma_{23})$

where $s = ((\sigma_{00}^2\sigma_{13}^2 - 2\sigma_{00}\sigma_{01}\sigma_{03}\sigma_{13} + \sigma_{01}^2\sigma_{03}^2)\sigma_{24}^2 + (((2\sigma_{00}\sigma_{01}\sigma_{03} - 2\sigma_{00}^2\sigma_{13})\sigma_{14} + 2\sigma_{00}\sigma_{01}\sigma_{04}\sigma_{13} - 2\sigma_{01}^2\sigma_{03}\sigma_{04})\sigma_{23} + (2\sigma_{00}\sigma_{02}\sigma_{03}\sigma_{13} - 4\sigma_{00}\sigma_{03}^2\sigma_{12} + 2\sigma_{01}\sigma_{02}\sigma_{03}^2)\sigma_{14} - 2\sigma_{00}\sigma_{02}\sigma_{04}\sigma_{13}^2 + (4\sigma_{00}\sigma_{03}\sigma_{04}\sigma_{12} - 2\sigma_{01}\sigma_{02}\sigma_{03}\sigma_{04})\sigma_{13})\sigma_{24} + (\sigma_{00}^2\sigma_{14}^2 - 2\sigma_{00}\sigma_{01}\sigma_{04}\sigma_{14} + \sigma_{01}^2\sigma_{04}^2)\sigma_{23}^2 + (-2\sigma_{00}\sigma_{02}\sigma_{03}\sigma_{14}^2 + (2\sigma_{00}\sigma_{02}\sigma_{04}\sigma_{13} + 4\sigma_{00}\sigma_{03}\sigma_{04}\sigma_{12} - 2\sigma_{01}\sigma_{02}\sigma_{03}\sigma_{04})\sigma_{14} + (2\sigma_{01}\sigma_{02}\sigma_{04}^2 - 4\sigma_{00}\sigma_{04}^2\sigma_{12})\sigma_{13})\sigma_{23} + \sigma_{02}^2\sigma_{03}^2\sigma_{14}^2 - 2\sigma_{02}^2\sigma_{03}\sigma_{04}\sigma_{13}\sigma_{14} + \sigma_{02}^2\sigma_{04}^2\sigma_{13}^2)$

The solutions are distinct because $[\sqrt{s}]_{\mathcal{G}}$ simplifies to non-zero $\sqrt{{\omega_{15}}^2(\omega_{13}\omega_{24} + 2\lambda_{13}\lambda_{34}\omega_{12}\omega_{13} + \omega_{33}\,\lambda_{34}\omega_{12})^2}$.

Only the first solution is valid for the 4-cycle $1 \leftrightarrow 2 \leftrightarrow 3 \leftrightarrow 4$, so the graph is fully identifiable.

B.2.4 Identification of (360, 369)



There are three missing cycles $1 \leftrightarrow 2 \leftrightarrow 3$, $1 \leftrightarrow 3 \leftrightarrow 4$, and $1 \leftrightarrow 2 \leftrightarrow 3 \leftrightarrow 4$. The missing cycle $1 \leftrightarrow 2 \leftrightarrow 3$ gives two solutions:

$\lambda_1 = (\sqrt{s} + (\sigma_{00}\sigma_{12} + \sigma_{01}\sigma_{02})\sigma_{23} + (-\sigma_{00}\sigma_{13} - \sigma_{01}\sigma_{03})\sigma_{22} + \sigma_{02}^2\sigma_{13} - \sigma_{02}\sigma_{03}\sigma_{12})/(2\sigma_{00}\sigma_{02}\sigma_{23} - 2\sigma_{00}\sigma_{03}\sigma_{22})]$, and

$\lambda_1' = (-\sqrt{s} + (\sigma_{00}\sigma_{12} + \sigma_{01}\sigma_{02})\sigma_{23} + (-\sigma_{00}\sigma_{13} - \sigma_{01}\sigma_{03})\sigma_{22} + \sigma_{02}^2\sigma_{13} - \sigma_{02}\sigma_{03}\sigma_{12})/(2\sigma_{00}\sigma_{02}\sigma_{23} - 2\sigma_{00}\sigma_{03}\sigma_{22})$

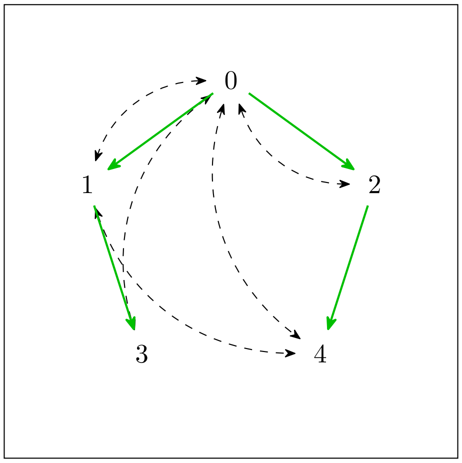
where $s = (\sigma_{00}^2\sigma_{12}^2 - 2\sigma_{00}\sigma_{01}\sigma_{02}\sigma_{12} + \sigma_{01}^2\sigma_{02}^2)\sigma_{23}^2 + (((2\sigma_{00}\sigma_{01}\sigma_{02} - 2\sigma_{00}^2\sigma_{12})\sigma_{13} + 2\sigma_{00}\sigma_{01}\sigma_{03}\sigma_{12} - 2\sigma_{01}^2\sigma_{02}\sigma_{03})\sigma_{22} + (2\sigma_{01}\sigma_{02}^3 - 2\sigma_{00}\sigma_{02}^2\sigma_{12})\sigma_{13} + 2\sigma_{00}\sigma_{02}\sigma_{03}\sigma_{12}^2 - 2\sigma_{01}\sigma_{02}^2\sigma_{03}\sigma_{12})\sigma_{23} + (\sigma_{00}^2\sigma_{13}^2 - 2\sigma_{00}\sigma_{01}\sigma_{03}\sigma_{13} + \sigma_{01}^2\sigma_{03}^2)\sigma_{22}^2 +$

$(-2\sigma_{00}\sigma_{02}^2\sigma_{13}^2 + (6\sigma_{00}\sigma_{02}\sigma_{03}\sigma_{12} - 2\sigma_{01}\sigma_{02}^2\sigma_{03})\sigma_{13} - 4\sigma_{00}\sigma_{03}^2\sigma_{12}^2 + 2\sigma_{01}\sigma_{02}\sigma_{03}^2\sigma_{12})\sigma_{22} + \sigma_{02}^4\sigma_{13}^2 - 2\sigma_{02}^3\sigma_{03}\sigma_{12}\sigma_{13} + \sigma_{02}^2\sigma_{03}^2\sigma_{12}^2$.

The solutions are distinct because $[\sqrt{s}]_{\mathcal{G}}$ simplifies to non-zero $\sqrt{{\omega_{01}}^2(2\lambda_{02}\omega_{02} + \omega_{22})^2{\omega_{33}}^2}$.

If we insert this in the cycle $1 \leftrightarrow 2 \leftrightarrow 3 \leftrightarrow 4$, maxima says neither is a solution because there are terms involving $\mid \omega_{03} \mid$ that do not cancel. Manually replacing $\mid \omega_{03} \mid$ by (ω_{03}) discovers that only the first λ_1 is a solution. Thus the graph is fully identifiable. (alternative the λ_1 from cycle 134 works without manual replacement)

B.2.5 Identification of (840, 466)



There are three missing cycles $1 \leftrightarrow 2 \leftrightarrow 3$, $2 \leftrightarrow 3 \leftrightarrow 4$, and $2 \leftrightarrow 1 \leftrightarrow 3 \leftrightarrow 4$. The missing cycle $1 \leftrightarrow 2 \leftrightarrow 3$ gives two solutions:

$\lambda_1 = (\sqrt{s} + (\sigma_{00}\sigma_{11} + \sigma_{01}^2)\sigma_{23} + (\sigma_{01}\sigma_{02} - \sigma_{00}\sigma_{12})\sigma_{13} - \sigma_{01}\sigma_{03}\sigma_{12} - \sigma_{02}\sigma_{03}\sigma_{11})/(2\sigma_{00}\sigma_{01}\sigma_{23} - 2\sigma_{00}\sigma_{03}\sigma_{12})$, and

$\lambda_1' = -(\sqrt{s} + (-\sigma_{00}\sigma_{11} - \sigma_{01}^2)\sigma_{23} + (\sigma_{00}\sigma_{12} - \sigma_{01}\sigma_{02})\sigma_{13} + \sigma_{01}\sigma_{03}\sigma_{12} + \sigma_{02}\sigma_{03}\sigma_{11})/(2\sigma_{00}\sigma_{01}\sigma_{23} - 2\sigma_{00}\sigma_{03}\sigma_{12})$

where $s = (\sigma_{00}^2\sigma_{11}^2 - 2\sigma_{00}\sigma_{01}^2\sigma_{11} + \sigma_{01}^4)\sigma_{23}^2 + (((-2\sigma_{00}^2\sigma_{11} - 2\sigma_{00}\sigma_{01}^2)\sigma_{12} + 2\sigma_{00}\sigma_{01}\sigma_{02}\sigma_{11} + 2\sigma_{01}^3\sigma_{02})\sigma_{13} + (6\sigma_{00}\sigma_{01}\sigma_{03}\sigma_{11} - 2\sigma_{01}^3\sigma_{03})\sigma_{12} - 2\sigma_{00}\sigma_{02}\sigma_{03}\sigma_{11}^2 - 2\sigma_{01}^2\sigma_{02}\sigma_{03}\sigma_{11})\sigma_{23} + (\sigma_{00}^2\sigma_{12}^2 - 2\sigma_{00}\sigma_{01}\sigma_{02}\sigma_{12} + \sigma_{01}^2\sigma_{02}^2)\sigma_{13}^2 + (2\sigma_{00}\sigma_{01}\sigma_{03}\sigma_{12}^2 + (2\sigma_{00}\sigma_{02}\sigma_{03}\sigma_{11} - 2\sigma_{01}^2\sigma_{02}\sigma_{03})\sigma_{12} - 2\sigma_{01}\sigma_{02}^2\sigma_{03}\sigma_{11})\sigma_{13} + (\sigma_{01}^2\sigma_{03}^2 - 4\sigma_{00}\sigma_{03}^2\sigma_{11})\sigma_{12}^2 + 2\sigma_{01}\sigma_{02}\sigma_{03}^2\sigma_{11}\sigma_{12} + \sigma_{02}^2\sigma_{03}^2\sigma_{11}^2$.

The solutions are distinct because $[\sqrt{s}]_{\mathcal{G}}$ simplifies to non-zero $\sqrt{(2\lambda_{01}\omega_{01} + \omega_{11})^2{\omega_{02}}^2{\omega_{03}}^2}$.

If we insert this in the cycle $1 \leftrightarrow 3 \leftrightarrow 4 \leftrightarrow 2$, maxima says neither is a solution because there are terms involving $\mid \omega_{02} \mid$ and $\mid \omega_{03} \mid$ that do not cancel. Manually replacing them with (ω_{02}) and (ω_{03}) discovers that only the first λ_1 is a solution. Thus the graph is fully identifiable.

B.3 Canonical path graphs

For Proposition 5, we have calculated the identifiability of all canonical path graphs with exactly one missing cycle of length at most 10 using Gröbner bases.

The results are shown in the file `canonical-cycles.pdf`.

The directed edges form a path $0 \to 1 \to 2 \to 3 \to \ldots$ in each graph.

The color of the directed edges encodes the identifiability:

1. green: The edge is uniquely identifiable.
2. yellow: The edge is 2-identifiable.
3. blue: The Gröbner base does not include a solution the edge identifiability directly, but the edge can be

identified using some kind of propagate from adjacent edges. It is either 1-identifiable or 2-identifiable depending on the color of the adjacent edge.

4. black: The edge is not identifiable

The number of nodes is not specified, since additional nodes do not affect the identifiability

The graphs are normalized for shifting, but not for permutations. E.g. the 2nd and 3rd graph of length 3 in `canonical-cycles.pdf` are equivalent under permutations, and so must have the same identifiability. Hence here one can see more fully identifiable graphs of length 4 than in the main paper, which is not a contradiction, as they are equivalent.

B.4 Comparison with Gröbner bases

Besides path graphs with a single missing cycle, we have enumerated 879 path graphs with 8 nodes and various combinations of bidirected edges. On these graphs, we have searched the identifiable edges with TreeID and a Gröbner base approach. The results are shown in the pdf files of the folder `879graphs`.

TreeID (as implemented in DAGitty) completed its computation in a day and night on a laptop. To calculate the Gröbner bases, we have used Singular (Decker et al., 2016). Over several months on a desktop PC, it calculated the Gröbner bases for the first 587 graphs and then it stopped proceeding, possibly having exhausted the available RAM. So we have aborted it, and continued the computations for the remaining graphs with a time limit of 4 hours per graph on a server. There it eventually finished.

Comparing the output of both algorithms, we see that TreeID can identify all edges that can be identified using Gröbner bases on these graphs.

Furthermore, it can identify edges that could not be identified with the Gröbner bases, which should be impossible. This seems to occur due to two bugs in our Gröbner base analysis: 1) when the 4 hour time limit was breached, we recorded all edges as unidentifiable rather than a computation failure, and 2) prioritizing base polynomials containing a single variable over propagation. For example, the Gröbner base might contain $p = \lambda_2^2\Sigma_1 + \lambda_2\Sigma_2 + \Sigma_3$, $q = \lambda_2\Sigma_4 + \lambda_1\Sigma_5 + \Sigma_6$ and $r = \lambda_1\Sigma_7 + \Sigma_8$, where Σ_i are arbitrary polynomials in $\sigma_{j,k}$. From p, our analysis script would conclude that λ_2 is 2-identifiable, and from r that λ_1 is 1-identifiable. Our script would then stop, assuming it had discovered the identifiability of all edges. However, from q and r together, one can conclude that λ_2 is also 1-identifiable. As explained in the previous section, in our visualizations we have drawn edges identified by p, q, resp. r as yellow, blue, resp. green. Essentially the bug was to draw edges that might be blue or yellow as yellow, even if blue was better.

B.5 wxMaxima files

We have performed several calculations in the wxMaxima CAS. For the sake of reproducibility, we share the following wxMaxima files:

1. `3nodes.wxmx`, `5nodes.wxmx`, `binaryTree.wxmx`, `binaryTree2.wxmx`: Examples of the calculation of the Σ matrix.

2. `propagate.wxmx`: Shows that the propagation step of a FASTP results in a FASTP.

3. `recursion.wxmx` and `recursion-abcd.wxmx`: A general recursion scheme for a cycle of 3,4, and 6 length. In the first level, the a, b, c, d have not been replaced by any σ_{ij}, so the recursion returns a general quadratic equation. In this general quadratic equation, the a, b, c, d can be replaced by σ_{ij} to obtain a quadratic equation for a specific system without performing the recursion.

4. `substitution-scheme-3-equations.wxmxm`, `substitution-scheme-4-equations.wxmx`: An alternative way of solving the equation system of a missing cycle using substitution rather than the recursion.

5. `substitution-abcd.wxmxm`: Shows that substitution unlike the recursion does not work in general.

6. `test-drton-tsiv-*.wxmx`: The calculations for Figure 3.

7. `test-drton-tsiv-Fig2*.wxmx`: The calculations for $\mathcal{G}_2$ in Figure 1 (which is a graph in Figure 2 in Weihs et al. (2018)).

8. `test-discriminant-cycle*.wxmx`: The calculations for Figure 4 (canonical path graphs with no solutions or a unique solution).

B.6 Script files

In the folder `scripts`, you can find some of the code we have used to run the experiments.

1. `singlecyclepathgraphs.lpr`: Creates path graphs with a single missing cycle.

   Example usage after compilation:
   ```
   ./singlecyclepathgraphs 4
   ```

2. `identification-helper.xqm`: Various functions used by the other scripts, mostly to convert graphs from one file format to another. The functions can also be called directly:

   Example:
   ```
   xidel --module identification-helper.xqm -e ’helper:drton-model-to-pretty-edge-string(5, "(4680, 403)")’
   ```
   to convert the graph $(4680, 403)$ in the notation of Weihs et al. (2018) to our format.

3. `drtonModelToGraph.lpr`: Another program to convert the graphs of Weihs et al. (2018) to our format. The input graph is specified as constants in the source code.

4. `identifiability-singular-model.xq`: Creates Singular commands to calculate the Gröbner base for some graphs. It outputs variables `L`i for λ_i and `s`i`s`j for σ_{ij}.

   Example:
   ```
   echo ’[[[1, 2], [1, 4], [2,3], [2,4], [3,4]]]’ | xidel --input-format json - -e @identifiability-singular-model.xq | Singular
   ```

5. `identifiable-iff.compress.pl`: Removes all sigma variables from the Gröbner base output of Singular to save space.

6. `identifiable-iff.parsesingular.pl`: Parses the Gröbner base output to find the identifiable edges (see B.3 and B.4).

   Example:
   ```
   Singular < input-with-the-singular-model | perl identifiable-iff.compress.pl | perl identifiable-iff.parsesingular.pl > output.tex
   ```

   The output consists of TeX commands which can create the visualizations in the folder `879graphs` when combined with suitable definitions for the commands.

7. `makegraph-results-to-json.xq`: This converts the output of `identifiable-iff.parsesingular.pl` to JSON. The JSON can be copied into a JavaScript program, from which it is easy to call DAGitty.

8. `identifiable-iffgraphs-cycles-solution.xq`: Calculates the quadratic equation of Theorem 1 for a given graph and missing cycle using the recursion of Definition 2.

   Example:
   ```
   model=’1->2 1->3 1->4 4->5 1<->2 1<->3 1<->4 1<->5’ cycle=’2 3 4’ xidel identifiable-iffgraphs-cycles-solution.xq
   ```

   The last three lines of the output can be copied verbatim into Maxima to define the variables a, b, c for the quadratic equation. This is the script we have used to calculate the solutions in B.1 and B.2, in combination with the next two scripts which reveal how many solutions the quadratic equation has.

9. `graph-to-matrices.xq`: Creates Maxima commands to calculate Λ, Ω, Σ matrices for a given graph.

   Example:
   ```
   model=’[[1, 2], [2, 3], [1, 3]]’ xidel graph-to-matrices.xq
   ```

10. `discriminant.xq`: Creates Maxima commands to substitute the elements of a Σ matrix of size `n` into the expressions a and $b^2 - 4ac$ of Lemma 7.

    Example:
    ```
    n=5 xidel discriminant.xq
    ```

`*.pl` files are run with Perl, `*.xq` files are run with Xidel, `*.lpr` files are compiled with FreePascal.

There are two different formats used to read to graphs in the scripts. A JSON syntax that only lists the missing bidirected edges of a graph (e.g. `[[1,2]]` for a missing edge $1 \leftrightarrow 2$) and a format that lists all existing edges (e.g. `1->2 2<->3`). Some scripts assume the root node is node 1 (especially those creating commands for Maxima and Singular), some scripts assume the root node is node 0. The user needs to pay attention to this when using the scripts.