On the Complexity of Counterfactual Reasoning

Anonymous Author(s)
Affiliation
Address
email

Abstract

A common form of counterfactual reasoning is based on the notion of *twin network* which is a causal graph that represents two worlds, one real and another imaginary. Information about the real world is used to update the joint distribution over the network's causal mechanisms which is then used for hypothetical reasoning in the imaginary world. This is in contrast to associational and interventional reasoning which involve a causal graph over a single world that we shall call a *base network*. In this paper, we study the complexity of counterfactual reasoning in twin networks in relation to the complexity of associational and interventional reasoning in base networks. We show that counterfactual reasoning is no harder than associational/interventional reasoning in the context of two computational frameworks. One of these is based on the notion of *treewidth* and includes the classical variable elimination and jointree algorithms. The second, more recent framework is based on the notion of *causal treewidth* which is directed towards causal graphs used in counterfactual reasoning (e.g., structural causal models). More specifically, we show that the (causal) treewidth of a twin network is at most twice the (causal) treewidth of its base network plus one. This means that if associational/interventional reasoning is tractable then counterfactual reasoning is also tractable. We further extend our results to a form of counterfactual reasoning that is more general than what is commonly discussed in the literature. We finally present empirical results that measure the gap between the complexities of counterfactual reasoning and associational/interventional reasoning on random networks.

1 Introduction & Background

A theory of causality has emerged over the last few decades based on two parallel hierarchies, an *information hierarchy* and a *reasoning hierarchy,* often called the *causal hierarchy* [21]. On the reasoning side, this theory has crystalized three levels of reasoning with increased sophistication and proximity to human reasoning: associational, interventional and counterfactual, which are exemplified by the following canonical probabilities. *Associational* $Pr(y|x)$: probability of y given that x was observed. Example: probability that a patient has a flu given they have a fever. *Interventional* $Pr(y_x)$: probability of y given that x was established by an intervention. This is different from $Pr(y|x)$. Example: seeing the barometer fall tells us about the weather but moving the barometer needle won't bring rain. *Counterfactual* $Pr(y_x|\bar{y}, \bar{x})$: probability of y if we were to establish x by an intervention given that neither x nor y are true. Example: probability that a patient who did not take a vaccine and died would have recovered had they been vaccinated. On the information side, these forms of reasoning were shown to require different levels of knowledge, encoded as (1) associational models, (2) causal models and (3) functional (mechanistic) models, respectively, with each class of models containing more information than the preceding one. In the framework of probabilistic graphical models [17], this information is encoded by (1) Bayesian networks [7, 19], (2) causal Bayesian networks [20, 22, 26], and (3) functional Bayesian networks [3, 20].

Submitted to *A causal view on dynamical systems* workshop at NeurIPS 2022. Do not distribute.

Counterfactual reasoning has received much interest historically as it inspires both introspection and contemplating scenarios that have not been seen before, and is therefore viewed by many as a hallmark of human intelligence. Figure 1 depicts a functional Bayesian network, also known as a *structural causal model (SCM)*, which can be used to answer counterfactual queries. Variables without causes are called *exogenous* or *root* and variables with causes are called *endogenous* or *internal.* The only uncertainty in SCMs concerns the states of exogenous variables and this uncertainty is quantified using distributions over these variables. Endogenous variables are assumed to be *functional:* they are functionally determined by their causes where the functional relationships, also known as *causal mechanisms,* are specified by equations.[1] These equations and the distributions over exogenous variables define the *parameters* of the causal graph, leading to a fully specified model which can be used to evaluate associational, interventional and counterfactual queries. For example, the SCM in Figure 1 has enough information to evaluate the counterfactual query $Pr(y_x|\bar{x}, \bar{y})$: the probability that a patient who did not take the treatment and died would have been alive had they been given the treatment. A causal Bayesian network contains less information than a functional one (SCM) as it does not require endogenous variables to be functional, but it is sufficient to compute associational and interventional probabilities. A Bayesian network contains even less information as it does not require network edges to have a causal interpretation, only that the conditional independences encoded by the network are correct, so it can only compute associational probabilities.

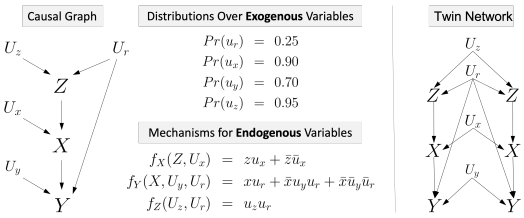


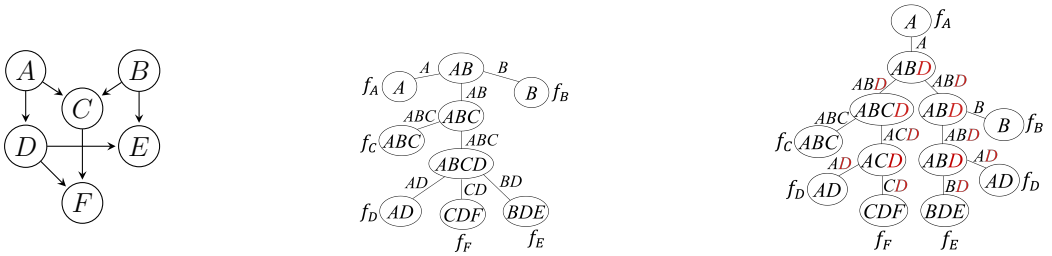
Figure 1: A structural causal model [4] and its twin network. Endogenous variables represent treatment (X), the outcome of (Y), and the presence of (Z), hypertension. Exogenous variables represent natural resistance to disease (U_r) and sources of variation affecting endogenous variables (U_x, U_y, U_z).

All three forms of reasoning (and models) involve a directed acyclic graph (DAG) which we call the *base network.* The time complexity of associational and interventional reasoning can be bounded by $n \cdot \exp(w)$ where n is the number of network nodes and w is its treewidth (a graph-theoretic measure of connectivity). Counterfactual reasoning is based on a *twin network* which is obtained by duplicating internal nodes in the base network; see right of Figure 1. To compute the counterfactual query $Pr(y_x|\bar{y}, \bar{x})$, one asserts $\bar{y}, \bar{x}$ as an observation on one side of the twin network (real world) and computes the interventional query $Pr(y_x)$ on the other side of the network (imaginary world). One can also bound the time complexity of reasoning by $n^t \cdot \exp(w^t)$ where n^t is the number of nodes in the twin network and w^t is its treewidth. A recent result provides a similar but much tighter bound using the notion of *causal treewidth* [5, 9], which is no greater than treewidth but applies only when certain nodes in the base network are functional — in SCMs every internal node is functional.

One would expect the more sophisticated counterfactual reasoning with twin networks to be more expensive than associational/interventional reasoning with base networks since the former networks are larger and have more complex topologies. But the question is: How much more expensive? For example, can associational/interventional reasoning be tractable on a base network when counterfactual reasoning is intractable on its twin network? We address this question in the context of reasoning algorithms whose complexity is exponential only in the (causal) treewidth, such as the jointree algorithm [18], the variable elimination algorithm [29, 12] and circuit-based algorithms [6, 10]. In particular, we show that the (causal) treewidth of a twin network is at most twice the (causal) treewidth of its base network plus one. This means that the complexity of counterfactual reasoning is no more than quadratic in the complexity of associational and interventional reasoning so the former must be bounded if the latter is bounded. We also extend our results to a more general form of counterfactual reasoning which requires contemplating more than two worlds, therefore demanding networks that are more complex than twin networks. We finally present empirical results which reveal that, on average, the complexity gap between counterfactual and associational/interventional reasoning is smaller than what our worst-case bounds may suggest.

Our results apply not only to computing queries using fully specified models but also to estimating queries from data and an unparameterized base network. The classical approach for this task is to derive an estimand for the query when it is identifiable; see [20] for a detailed exposure of this

[1] These equations can also be specified using conditional probability tables (CPTs) that are normally used in Bayesian networks, but the CPTs will contain only deterministic distributions.



(a) DAG (b) jointree (width 3) (c) thinned jointree (width 2)

Figure 2: A family f appears next to a jointree node i iff f is hosted by i ($i \in \mathcal{H}(f)$). The family of variable D is replicated in (c) since $|\mathcal{H}(f_D)| = 2$, where red variables are thinned, assuming that variable D is functional.

approach. A more recent class of approaches first estimate the parameters of the base network, leading to a fully specified model, or set of models, which are then used to compute the query using classical reasoning algorithms; see, e.g., [28, 27, 9].[2] These approaches lead to point estimates when the query is identifiable and to bounds for queries that are not identifiable. The complexity of estimating parameters and the following reasoning process can both be bounded using (causal) treewidth, therefore benefiting from the positive results we present in this paper.

2 Technical Preliminaries

We next review the notions of treewidth [23] and causal treewidth [5, 9, 8] which we use to characterize the computational complexity of counterfactual reasoning. We also review the of elimination orders, jointrees and thinned jointrees which are the basis for defining (causal) treewidth and characterizing the computational complexity of various reasoning algorithms. We use these notions extensively when stating and proving our results (proofs of all results are in the Appendix). We denote variables by uppercase letters (e.g. X) and their values by lowercase letters (e.g. x). A set of variables is denoted by a bold uppercase letter (e.g. $\mathbf{X}$) and its instantiations by a bold lowercase letter (e.g. $\mathbf{x}$).

2.1 Elimination Orders

These are total orders of the network variables which drive, and characterize the complexity of, the classical variable elimination algorithm when computing associational, interventional and counterfactual queries. Consider a DAG G where every node represents a variable. An *elimination order* π for G is a total ordering of the variables in G, where $\pi(i)$ is the ith variable in the order, starting from $i = 1$. An elimination order defines an elimination process on the moral graph of DAG G which is used to define the treewidth of G. The *moral graph* G_m is obtained from G by adding an undirected edge between every pair of common parents and then removing directions from all directed edges. When we *eliminate* variable $\pi(i)$ from G, we connect every pair of neighbors of $\pi(i)$ in G_m and remove $\pi(i)$ from G_m. This elimination process induces a *cluster sequence* $\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_n$, where $\mathbf{C}_i$ is $\pi(i)$ and its neighbors in G_m just before eliminating $\pi(i)$. The *width* of an elimination order is the size of its largest induced cluster minus 1. The *treewidth* for DAG G is the minimum width of any elimination order for G. Elimination orders are usually constructed using heuristics that aim to minimize their width. The *minfill* heuristic [16] is perhaps the most effective and popular. The variable elimination algorithm computes queries in $O(n \cdot \exp(w))$ time where n is the number of nodes in the (base or twin) network and w is the width of a corresponding elimination order.

2.2 Jointrees

These are data structures that drive, and characterize the complexity of, the classical jointree algorithm; see Figure 2b. Let the *family* of variable X in DAG G be the set f_X containing X and its parents in G. A *jointree* for DAG G is a pair $\langle \mathcal{T}, \mathcal{H} \rangle$ where $\mathcal{T}$ is a tree and $\mathcal{H}$ is a function that maps each family f of G into nodes $\mathcal{H}(f)$ in $\mathcal{T}$ called the *hosts* of family f. The requirements are: only nodes

[2] Another recent direction is to reduce the estimation of queries to a constrained optimization problem [13].

with a single neighbor (called *leaves*) can be hosts; each leaf node hosts exactly one family; and each family must be hosted by at least one node.[3] This induces a *cluster* $\mathbf{C}_i$ for each jointree node i and a *separator* $\mathbf{S}_{ij}$ for each jointree edge (i, j) which are defined as follows. Separator $\mathbf{S}_{ij}$ is the set of variables hosted at both sides of edge (i, j). If jointree node i is a leaf then cluster $\mathbf{C}_i$ is the family hosted by i; otherwise, $\mathbf{C}_i$ is the union of separators adjacent to node i. The *width* of a jointree is the size of its largest cluster minus 1. The minimum width of all jointrees for G corresponds to the treewidth of G. There are polytime, width-preserving transformations between elimination orders and jointrees; see [7, Ch 9] for details. Jointrees are usually constructed from minfill elimination orders. The jointree algorithm computes queries in $O(n \cdot \exp(w))$ time where n is the number of nodes and w is the width of a corresponding jointree.

2.3 Thinned Jointree

To *thin* a jointree is to remove some variables from its separators (and hence clusters, which are defined in terms of separators); see Figure 2c. Thinning reduces the jointree width, sometimes quite significantly, leading to exponential savings in reasoning time. Thinning is possible only when some variables in the network are functional, even without knowing the specific functional relationships (i.e., structural equations). The *causal treewidth* is the minimum width for any thinned jointree. Causal treewidth is no greater than treewidth and the former can be bounded when the latter is not. While this notion can be applied broadly as in [8], it is particularly relevant to counterfactual reasoning since every internal node in an SCM is functional so the causal treewidth for these models can be significantly smaller than their treewidth. There are alternate definitions of thinned jointrees. The next definition is based on thinning rules [5].

A thinning rule removes a variable from a separator under certain conditions. There are two thinning rules which apply only to functional variables. The first rule removes variable X from a separator $\mathbf{S}_{ij}$ if edge (i, j) is on the path between two leaf nodes that host the family of X and every separator on that path contains X. The second rule removes variable X from a separator $\mathbf{S}_{ij}$ if no other separator $\mathbf{S}_{ik}$ contains X, or no other separator $\mathbf{S}_{kj}$ contains X. A thinned jointree is obtained by applying these rules to exhaustion. Figure 2 depicts an optimal, classical jointree and a thinned jointree for the same DAG (the latter has smaller width).

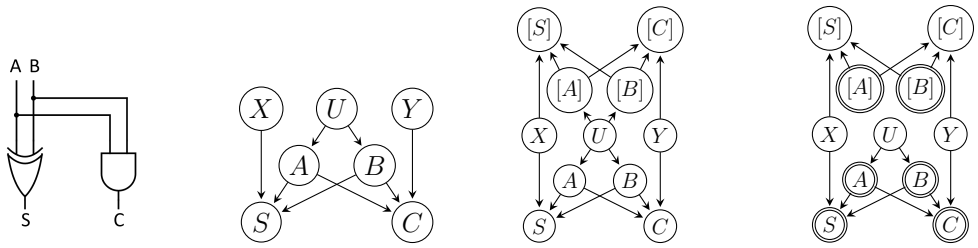
The effectiveness of thinning rules depends on the number of jointree nodes that host a family f, $|\mathcal{H}(f)|$, and the location of these nodes in the jointree. One can enable more thinnings by increasing the number of jointree nodes that host each family f. This process is called *replication* where $|\mathcal{H}(f)|$ is called the number of *replicas* for family f. Replication comes at the expense of increasing the number of jointree nodes so the definition of causal treewidth limits this growth by requiring the jointree size to be a polynomial in the number of nodes in the underlying DAG; see [5] for details.[4]

3 The Treewidth of Twin Networks

Consider Figure 3a which depicts a 2-bit half adder. Suppose the binary inputs A and B are randomly sampled from some distribution and the gates may not be functioning properly. This circuit can be modeled using the network in Figure 3b. Variables A, B, S, C represent the inputs and outputs of the circuit; X, Y represent the health of the XOR gate and the AND gate; and U represents an unknown external random sampler that decides the state of inputs A and B. Suppose that currently input A is high, input B is low yet both outputs C and S are low which is an abnormal circuit behavior. We wish to know whether the half adder would still behave correctly when we turn both inputs A and B on. This question can be formulated using the following counterfactual query: $Pr((c, \bar{s})_{a,b}|a, \bar{b}, \bar{c}, \bar{s})$. This query can be answered using a twin network as shown in Figure 3c, where each non-root variable V has a duplicate $[V]$. The current evidence $a, \bar{b}, \bar{c}, \bar{s}$ is asserted on the variables A, B, C, S representing the real world and the interventional query $Pr((c, \bar{s})_{a,b})$ is

[3] The standard definition of jointrees allows any node to be a host of any number of families. Our definition facilitates the upcoming treatment and does not preclude optimal jointrees.

[4] Thinning rules will not trigger if families are not replicated ($|\mathcal{H}(f)| = 1$ for all f). Replication usually increases the width of a jointree from w to w_r with the goal of having thinning rules reduce width w_r to width $w_t < w \leq w_r$. The replication strategy may sometimes not be effective on certain networks, leading to $w < w_t \leq w_r$. See [5, 8] for some replication strategies. We later use the one in [5] which exhibits this behavior on some networks.



(a) 2-bit half adder (b) base network (c) twin network (d) mutilated twin network

Figure 3: Internal nodes in the base network (Figure (b)) are functional. Figure (c),(d): Computing counterfactual $Pr((c, \bar{s})_{a,b}|a, \bar{b}, \bar{c}, \bar{s})$ using a twin network. Double-circled nodes have evidence.

computed on the duplicate variables $[A], [B], [C], [S]$ representing the imaginary world. This is done by removing the edges incoming into the intervened upon variables $[A], [B]$, asserting evidence $[a], [b]$ and finally computing the probability of $[c], [\bar{s}]$ as shown in Figure 3d; see [20] for an elaborate discussion of these steps. This basically illustrates how a counterfactual query can be computed using algorithms for associational queries, like variable elimination, but on a mutilated twin network instead of the base network.

We next show that the treewidth of a twin network is at most twice the treewidth of its base network plus one, which allows us to relate the complexities of assocational, interventional and counterfactual reasoning. First is the definition of twin networks as proposed by [2].

Definition 1. *Given a base network G, its twin network G^t is constructed as follows. For each internal variable X in G, add a new variable labeled $[X]$. For each parent P of X, if P is an internal variable, make $[P]$ a parent of $[X]$; otherwise, make P a parent of $[X]$. We will call X a base variable and $[X]$ a duplicate variable.*

For convenience, we use $[U] = U$ when U is root. For variables $\mathbf{X}$, we use $[\mathbf{X}]$ to denote $\{[X]|X \in \mathbf{X}\}$. Figure 3c depicts the twin network for the base network in Figure 3b.

3.1 Twin elimination orders

Our result on the treewidth of twin networks is based on converting every elimination order for the base network into an elimination order for its twin network while providing a guarantee on the width of the latter in terms of the width of the former. We provide a similar result for jointrees that we use when discussing the causal treewidth of twin networks.

Definition 2. *Consider an elimination order π for a base network G. The twin elimination order π^t is an elimination order for its twin network G^t constructed by replacing each non-root variable X in π by $X, [X]$.*

Consider the base network in Figure 3b and its elimination order $\pi = A, B, X, Y, S, C, U$. The twin elimination order will be $\pi^t = A, [A], B, [B], X, Y, S, [S], C, [C], U$.

Recall that eliminating variables $\pi(i), \ldots, \pi(n)$ from a base network G induces a cluster sequence $\mathbf{C}_1, \ldots, \mathbf{C}_n$. We use $\mathbf{C}(X)$ to denote the cluster of eliminated variable X. Similarly, eliminating variables from a twin network G^t induces a cluster sequence and we use $\mathbf{C}^t(X)$ to denote the cluster of eliminated variable X and $\mathbf{C}^t([X])$ to denote the cluster of its eliminated duplicate $[X]$.

Theorem 1. *Suppose we are eliminating variables from base network G using an elimination order π and eliminating variables from its twin network G^t using the twin elimination order π^t. For every variable X in G, we have $\mathbf{C}^t(X) \subseteq \mathbf{C}(X) \cup [\mathbf{C}(X)]$ and $\mathbf{C}^t([X]) \subseteq \mathbf{C}(X) \cup [\mathbf{C}(X)]$.*

This theorem has two key corollaries. The first relates the widths of an elimination order and its twin elimination order.

Corollary 1. *Let w be the width of elimination order π for base network G and let w^t be the width of twin elimination order π^t for twin network G^t. We then have $w^t \leq 2w + 1$.*

The above bound on w^t is tight as shown in the Appendix. Our major result is shown in Corollary 2.

Corollary 2. *If w is the treewidth of base network G and w^t is the treewidth of its twin network G^t, then $w^t \leq 2w + 1$.*

3.2 Twin jointrees

We will now provide a similar result for jointrees. That is, we will show how to convert a jointree $\langle \mathcal{T}, \mathcal{H} \rangle$ for a base network G into a jointree $\langle \mathcal{T}^t, \mathcal{H}^t \rangle$ for its twin network G^t while providing a guarantee on the width/size of the twin jointree in terms of the width/size of the base jointree. This may seem like a redundant result given Corollary 2 but the provided conversion will actually be critical for our later result on bounding the causal treewidth of twin networks. It can also be significantly more efficient than constructing a jointree by operating on the (larger) twin network.

Our conversion process operates on a *jointree* after directing its edges away from some node r, call it a *root.* This defines a single parent for each jointree node $i \neq r$, which is the neighbor of i closest to root r, with all other neighbors of i being its children. These parent-child relationships are invariant when running the algorithm. We also use a subroutine for *duplicating the jointree nodes rooted at some node i.* This subroutine duplicates node i and its descendant while also duplicating the edges connecting these nodes. If a duplicated node j hosts a family f, this subroutine will make $[j]$ host the duplicate family $[f]$ (so $j \in \mathcal{H}(f)$ iff $[j] \in \mathcal{H}([f])$).

The conversion process is given in Algorithm 1 which should be called initially with a root r that does not host a family for an internal DAG node and $p = null$. The twin jointree in Figure 4b was obtained from the base jointree in Figure 4a by this algorithm which simply adds nodes and edges to the base jointree. If an edge (i, j) in the base jointree is duplicated by Algorithm 1, we call (i, j) a *duplicated edge* and $([i], [j])$ a *duplicate edge.* Otherwise, we call (i, j) an *invariant edge.* In Figure 4b, duplicate edges are shown in red and invariant edges are shown in green. We now have the following key result on these twin jointrees.

Algorithm 1 Jointree to Twin Jointree

```
procedure MAKE-TWIN-JOINTREE(⟨T, H⟩, r, p)
    Σ ← leaf nodes at or below node r
    if nodes in Σ only host families for root variables then
        return
    if nodes in Σ only host families for internal variables then
        duplicate the jointree nodes rooted at node r
        add [r] as a child of p
    else
        for each child k of node r do
            MAKE-TWIN-JOINTREE(⟨T, H⟩, k, r)
```

Theorem 2. *If the input jointree to Algorithm 1 has separators $\mathbf{S}$ and the output jointree has separators $\mathbf{S}^t$, then for duplicated edges (i, j), $\mathbf{S}^t_{ij} = \mathbf{S}_{ij}$; for duplicate edges $([i], [j])$, $\mathbf{S}^t_{[i][j]} = [\mathbf{S}_{ij}]$; and for invariant edges (i, j), $\mathbf{S}^t_{ij} = \mathbf{S}_{ij} \cup [\mathbf{S}_{ij}]$.*

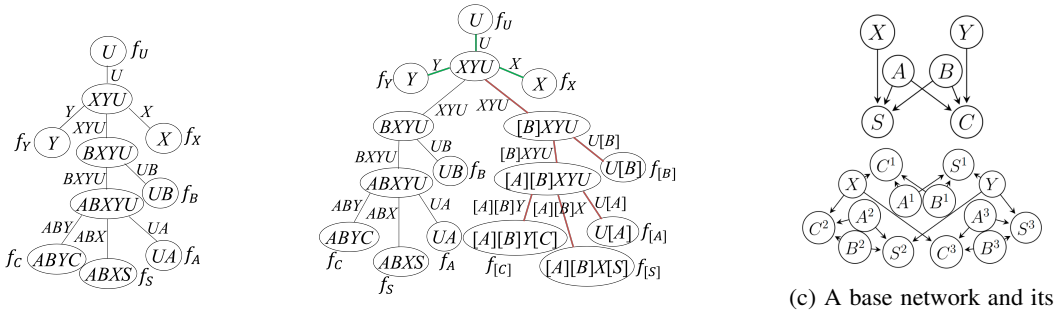
One can verify that the separators in Figure 4 satisfy these properties. The following result bounds the width and size of twin jointrees generated by Algorithm 1.

Corollary 3. *Let w be the width of a jointree for base network G and let n be the number of jointree nodes. Calling Algorithm 1 on this jointree will generate a jointree for twin network G^t whose width is no greater than $2w + 1$ and whose number of nodes is no greater than $2n$.*

The above bound on w^t is tight as shown in the Appendix. Since treewidth can be defined in terms of jointree width, the above result leads to the same guarantee of Corollary 2 on the treewidth of twin networks. However, the main role of the construction in this section is in bounding the causal treewidth of twin networks. This is discussed next.

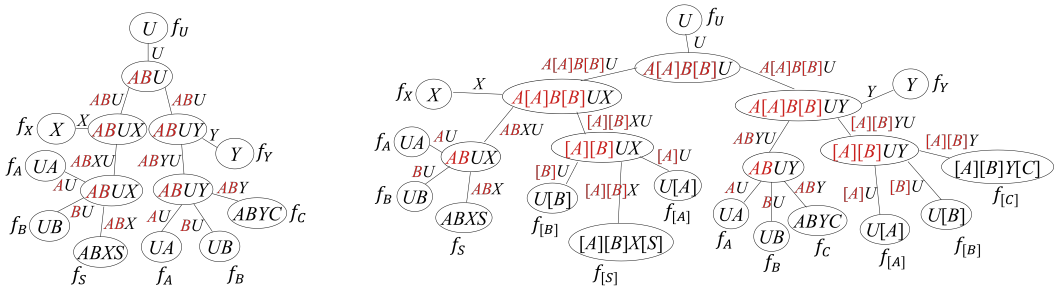
4 The Causal Treewidth of Twin Networks

Recall that causal treewidth is a more refined notion than treewidth as it uses more information about the network. In particular, this notion is relevant when we know that some variables in the network are functional, without needing to know the specific functions (equations) of these variables. By exploiting this information, one can construct thinned jointrees that have smaller separators and clusters compared to classical jointrees, which can lead to exponential savings in reasoning time [5, 9, 8]. As mentioned earlier, the causal treewidth corresponds to the minimum width of any thinned jointree. This is guaranteed to be no greater than treewidth and can be bounded when



(a) base jointree (b) twin jointree using Algorithm 1 (c) A base network and its 3-twin network

Figure 4: Figure (a),(b): A family f appears next to a jointree node i iff the family is hosted by that node ($i \in \mathcal{H}(f)$). Figure (c): an example of modeling using N-twin networks.



(a) thinned base jointree (width 3) (b) thinned twin jointree (width 3)

Figure 5: Illustrating the construction of a thinned, twin jointree from a thinned, base jointree.

treewidth is not [9]. We next show that the causal treewidth of a twin network is also at most twice the causal treewidth of its base network plus one. We start with the following key result.

Theorem 3. *Consider a twin jointree constructed by Algorithm 1 from a base jointree with thinned separators* $\mathbf{S}$*. The following are valid thinned separators for this twin jointree: for duplicated edges* (i, j)*,* $\mathbf{S}^t_{ij} = \mathbf{S}_{ij}$*; for duplicate edges* $([i], [j])$*;* $\mathbf{S}^t_{[i][j]} = [\mathbf{S}_{ij}]$*; and for invariant edges* (i, j)*,* $\mathbf{S}^t_{ij} = \mathbf{S}_{ij} \cup [\mathbf{S}_{ij}]$*.*

This theorem shows that a thinned, base jointree can be automatically converted into a thinned, twin jointree. This is significant for two reasons. First, this method avoids the explicit construction and thinning of jointrees for twin networks which can be quite expensive computationally [5]. Second, we have the following guarantee on the width of thinned, twin jointrees constructed by Theorem 3.

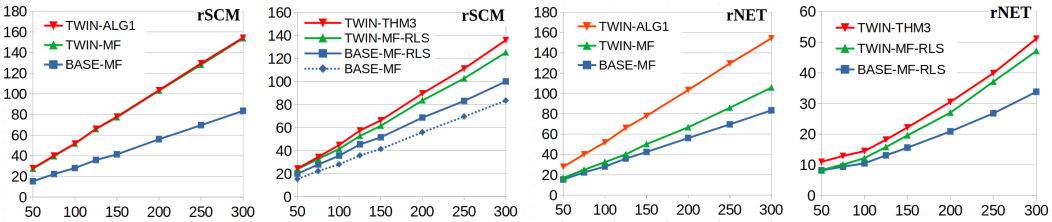
Corollary 4. *Consider the thinned, base and twin jointrees in Theorem 3. If the thinned, base jointree has width* w*, then the thinned, twin jointree has width no greater than* $2w + 1$*.*

Figure 5a shows a thinned jointree for the base network in Figure 3b. Figure 5b shows the corresponding thinned, twin jointree constructed by Algorithm 1 and annotated with the thinned separators (and clusters) as given by Theorem 3. We can now bound the causal treewidth of twin networks.

Corollary 5. *If* w *and* w^t *are the causal treewidths of a base network and its twin network, then* $w^t \leq 2w + 1$*.*

5 Beyond Twin Networks

Standard counterfactual reasoning contemplates two worlds, one real and another imaginary, while assuming that root nodes of the base network (exogenous variables) correspond to causal mechanisms that govern both worlds. This motivates the notion of a twin network as it ensures that these causal mechanisms are invariant. This type of reasoning can be thought of as a special case of *temporal reasoning* where variables change their states over time. In standard counterfactual reasoning, only



(a) classical jointrees (b) thinned jointrees (c) classical jointrees (d) thinned jointrees

Figure 6: Width of jointrees (y-axis) against number of base network nodes (x-axis) for maximum number of parents $p = 5$.

endogenous variables can change their states over time and we only have two time steps corresponding to the real and imaginary worlds.

A more general setup arises when we relax these assumptions. That is, we allow multiple time steps (worlds) and we allow some exogenous variables to also change their states over time. For an example of this, consider again the half adder in Figure 3a and its base network in Figure 4c. Suppose we set inputs A and B to high and low and observe outputs S and C to be high and low as well, which is a normal behavior. We then set both inputs to low and observe that the outputs do not change, which is an abnormal behavior. We then aim to predict the state of outputs if we were to set both inputs to high. This scenario involves three time steps (worlds). Moreover, while the health of gates X and Y are invariant over time, we do not wish to make the same assumption about the inputs A and B.

We can model this situation using the network in Figure 4c, which is not a twin network but a more general type of networks that appears quite frequently in practice. We call these N-twin networks.

Definition 3. *Consider a base network G and let $\mathbf{R}$ be a subset of its roots and $N \geq 1$ be an integer. The N-twin network G^N of G is constructed as follows. For each variable X in G that is not in $\mathbf{R}$, replace it with N duplicates of X, labeled $X^1, X^2, \ldots, X^N$. For each parent P of X, if P is in $\mathbf{R}$, make P a parent of X^i for all $i \in 1, 2, \ldots, N$. Else, make P^i a parent of X^i for all $i \in 1, 2, \ldots, N$.*

This definition generalizes the notion of a *parallel worlds model* [1] which falls as a special case when $\mathbf{R}$ contains all roots in the base network. Moreover, twin networks fall as a special case of this definition when $N = 2$ and $\mathbf{R}$ contains all roots of the base network. The question now is: can we provide guarantees on the (causal) treewidth of N-twin networks in terms of the (causal) treewidth of their base networks? The answer is yes as shown by the following generalization of earlier results.

Theorem 4. *If w and w^t are the (causal) treewidths of a base network and its N-twin network, $w^t \leq N(w + 1) - 1$.*

The class of N-twin networks can be viewed as a subclass of *dynamic Bayesian networks* [11] and is significant for a number of reasons. First, as illustrated by the above example, it arises when reasoning about the behavior of systems consisting of function blocks (e.g., gates) where each block has a health state that dictates its behavior and is invariant over time. This covers a large class of applications, particularly in model-based diagnosis [14]. Further, the class of N-twin networks allows counterfactual reasoning where conflicting observations and actions can arise in multiple worlds. Consider a treatment regime that is administered or not on a daily basis with outcomes also being collected daily. One may have observations and interventions that cover a number of days yet wishes to predict the result of an intervention on the next day. This can be achieved in a standard manner using N-twin networks. See [1, 24, 25] for additional, more sophisticated uses of N-twin networks.

6 Experimental Results

We consider experiments in this section that target random networks whose structures emulate the structures of SCMs used in counterfactual reasoning. We have a few objectives in mind. First, we wish to compare the widths of base and twin jointrees, with and without thinning. These widths do not correspond to (causal) treewidth since the jointrees are constructed using heuristics (finding optimal jointrees is NP-hard). Next, we want to compare the quality of twin jointrees constructed by Algorithm 1 (TWIN-ALG1), which operates directly on a base jointree, to the quality of twin jointrees

obtained by applying the minfill heuristic to a twin network (TWIN-MF). Recall that the former method is more efficient than the latter method. Finally, we wish to conduct a similar comparison between the thinned, twin jointrees constructed according to Theorem 3 (TWIN-THM3) and the thinned, twin jointrees obtained by applying the minfill heuristic and thinning rules to a twin network (TWIN-MF-RLS). Again, the former method is more efficient than the latter. The widths of these jointrees will be compared to the widths of base jointrees constructed by minfill (BASE-MF) and thinned, base jointrees constructed by minfill and thinning rules (BASE-MF-RLS).

We generated random networks according to the method used in [8]. Given a number of nodes n and a maximum number of parents p, the method chooses the parents of node X_i randomly from the set $X_1, \ldots, X_{i-1}$. The number of parents for node X_i is chosen randomly from the set $0, \ldots, min(p, i-1)$. We refer to these networks as rNET. We then consider each internal node N and add a unique root R as parent for N. This is meant to emulate the structure of SCMs as the exogenous variable R can be viewed as representing the different causal mechanisms for endogenous variable N. We refer to these modified networks as rSCM. The twin networks of rSCM are more complex than those for rNET since more variables are shared between the two slices representing the real and imaginary worlds (i.e., more information is shared between the two worlds).

We used $n \in \{50, 75, 100, 125, 150, 200, 250, 300\}$ and $p \in \{3, 5, 7\}$. For each combination of n and p, we generated 50 random, base networks and reported averages of two properties for the constructed jointrees: width and *normalized width.* If a jointree has clusters $\mathbf{C}_1, \ldots, \mathbf{C}_n$, then normalized width is $\log_2 \sum_{i=1}^{n} 2^{|\mathbf{C}_i|}$. This accounts for all clusters in the jointree instead of just the largest one (it also accounts for the jointree size). The data we generated occupies significant space so we included it in the Appendix while providing representative plots in Figure 6 for jointree widths under $p = 5$. We next discuss patterns exhibited in these plots and the full data in the Appendix.

First, the widths of twin jointrees are always less than twice the widths of their base jointrees and often significantly less than that. This is not guaranteed by our theoretical bounds as those apply to (causal) treewidth not to the widths of jointrees produced by heuristics — the latter widths are an upper bound on the former. Second, constructing a twin jointree by directly applying Algorithm 1 to a base jointree (TWIN-ALG1) is relatively comparable to constructing the twin jointree by operating on the twin network (TWIN-MF), as would normally be done. This also holds for thinned jointrees (TWIN-THM3 vs TWIN-MF-RLS) and is encouraging since the former methods are much more efficient than the latter ones. Third, the employment of thinned jointrees can lead to significant reduction in width and hence an exponential reduction in reasoning time. This can be seen by comparing the widths of twin jointrees TWIN-THM3 and TWIN-ALG1 since the former is thinned but the latter is not (similarly for TWIN-MF-RLS and TWIN-MF). Fourth, the twin jointrees of rSCM have larger widths than those of rNET. Recall that in rSCM, every endogenous variable has its own exogenous variable as a parent. Therefore, the distribution over exogenous variables has a larger space in rSCM compared to rNET. Since this distribution needs to be shared between the real and imaginary worlds, counterfactual reasoning with rSCM is indeed expected to be more complex computationally than reasoning with rNET. Finally, consider Figure 6b for a bottom-line comparison between the complexity of counterfactual reasoning and the complexity of associational/interventional reasoning in practice. Jointrees BASE-MF have the smallest widths for base networks so these are the jointrees one would use for associational/interventional reasoning. The best twin jointrees are TWIN-MF-RLS which are thinned. This is what one would use for counterfactual reasoning. The widths of latter jointrees are always less than twice the widths of the former, and quite often significantly much less.[5]

7 Conclusion

We studied the complexity of counterfactual reasoning using twin networks in relation to the complexity of associational and interventional reasoning. Our basic finding is that in the context of algorithms based on (causal) treewidth, the former complexity is no greater than quadratic in the latter. We extended these results to a form of counterfactual reasoning that goes beyond what is commonly discussed in the literature. Our empirical results suggest that for random SCMs, the complexity of counterfactual reasoning is closer to that of associational and interventional reasoning than our worst-case theoretical analysis may suggest.

[5] See footnote 4 for why BASE-MF is better than BASE-MF-RLS for rSCM but not for rNET.

References

[1] C. Avin, I. Shpitser, and J. Pearl. Identifiability of path-specific effects. In L. P. Kaelbling and A. Saffiotti, editors, *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, pages 357–363. Professional Book Center, 2005.

[2] A. Balke and J. Pearl. Probabilistic evaluation of counterfactual queries. In B. Hayes-Roth and R. E. Korf, editors, *Proceedings of the 12th National Conference on Artificial Intelligence, Seattle, WA, USA, July 31 - August 4, 1994, Volume 1*, pages 230–237. AAAI Press / The MIT Press, 1994.

[3] A. Balke and J. Pearl. Counterfactuals and policy analysis in structural models. In P. Besnard and S. Hanks, editors, *UAI '95: Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence, Montreal, Quebec, Canada, August 18-20, 1995*, pages 11–18. Morgan Kaufmann, 1995.

[4] E. Bareinboim, J. D. Correa, D. Ibeling, and T. F. Icard. On Pearl's hierarchy and the foundations of causal inference. 2021. Technical Report, R-60, Colombia University.

[5] Y. Chen and A. Darwiche. On the definition and computation of causal treewidth. In *UAI, Proceedings of the 38th Conference on Uncertainty in Artificial Intelligence*, 2022.

[6] A. Darwiche. A differential approach to inference in bayesian networks. *J. ACM*, 50(3):280–305, 2003.

[7] A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

[8] A. Darwiche. An advance on variable elimination with applications to tensor-based computation. In *ECAI*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 2559–2568. IOS Press, 2020.

[9] A. Darwiche. Causal inference with tractable circuits. In *Why-21 Workshop, NeurIPS*, 2021. https://arxiv.org/abs/2202.02891.

[10] A. Darwiche. Tractable Boolean and arithmetic circuits. In P. Hitzler and M. K. Sarker, editors, *Neuro-symbolic Artificial Intelligence: The State of the Art*, volume 342, chapter 6. Frontiers in Artificial Intelligence and Applications. IOS Press, 2022.

[11] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5, 1989.

[12] R. Dechter. Bucket elimination: A unifying framework for probabilistic inference. In *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 211–219, 1996.

[13] G. Duarte, N. Finkelstein, D. Knox, J. Mummolo, and I. Shpitser. An automated approach to causal inference in discrete settings. *CoRR*, abs/2109.13471, 2021.

[14] W. Hamscher, L. Console, and J. de Kleer, editors. *Readings in Model-Based Diagnosis*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992.

[15] J. S. Ide and F. G. Cozman. Random generation of bayesian networks. In G. Bittencourt and G. L. Ramalho, editors, *Advances in Artificial Intelligence, 16th Brazilian Symposium on Artificial Intelligence, SBIA 2002, Porto de Galinhas/Recife, Brazil, November 11-14, 2002, Proceedings*, volume 2507 of *Lecture Notes in Computer Science*, pages 366–375. Springer, 2002.

[16] U. Kjaerulff. Triangulation of graphs – algorithms giving small total state space. 1990. Technical Report R-90-09, Department of Mathematics and Computer Science, University of Aalborg, Denmark.

[17] D. Koller and N. Friedman. *Probabilistic Graphical Models - Principles and Techniques*. MIT Press, 2009.

[18] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 50(2):157–224, 1988.

[19] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[20] J. Pearl. *Causality*. Cambridge University Press, 2000.

[21] J. Pearl and D. Mackenzie. *The Book of Why: The New Science of Cause and Effect*. Basic Books, 2018.

[22] J. Peters, D. Janzing, and B. Schölkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. MIT Press, 2017.

[23] N. Robertson and P. D. Seymour. Graph minors. II. algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322, 1986.

[24] I. Shpitser and J. Pearl. What counterfactuals can be tested. In R. Parr and L. C. van der Gaag, editors, *UAI 2007, Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence, Vancouver, BC, Canada, July 19-22, 2007*, pages 352–359. AUAI Press, 2007.

[25] I. Shpitser and J. Pearl. Complete identification methods for the causal hierarchy. *J. Mach. Learn. Res.*, 9:1941–1979, 2008.

[26] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search, Second Edition*. Adaptive computation and machine learning. MIT Press, 2000.

[27] M. Zaffalon, A. Antonucci, and R. Cabañas. Causal expectation-maximisation. In *NIPS*, NeurIPS 2021 WHY-21 Workshop, 2021.

[28] M. Zaffalon, A. Antonucci, R. Cabañas, D. Huber, and D. Azzimonti. Bounding counterfactuals under selection bias. 2022.

[29] N. L. Zhang and D. L. Poole. Intercausal independence and heterogeneous factorization. In *UAI*, pages 606–614. Morgan Kaufmann, 1994.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [N/A]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Proofs

Proof of Theorem 1

Consider a base network G and its twin network G^t. We first introduce a set notation $\{X, [X]\}$, which contains both the base and duplicate variable for X if X is an internal variable and collapses to a single variable if X is a root variable.

For an elimination order π on a base network G, [7] defines a *graph sequence* $G_1, G_2, \ldots, G_n$ induced by π, where G_1 is the moral graph for G, and G_{i+1} is the result of eliminating $\pi(i)$ from G_i. Similarly, we define a *twin graph sequence* $G_1^t, G_2^t, \ldots, G_n^t$, where G_1^t is the moral graph for G^t, and G_{i+1}^t is the result of eliminating $\{\pi(i), [\pi(i)]\}$ from G_i^t.

Consider G_i in the graph sequence induced by π on G. For each variable X in G_i, let $G_i(X)$ be the set consisting of X and its neighbors in G_i. Similarly, let $G_i^t(X)$ and $G_i^t([X])$ denote the set consisting of X and its neighbors, and the set consisting of $[X]$ and its neighbors in G_i^t, respectively. By definition, $G_i(\pi(i)) = \mathbf{C}(\pi(i))$ and $G_i^t(\pi(i)) = \mathbf{C}^t(\pi(i))$.

We first propose a Lemma that relates $G_i^t(X)$ and $G_i(X)$.

Lemma 1. *Suppose we apply elimination order π to a base network G and apply π^t to its twin network G^t. Then for each variable X in G_i, we have $G_i^t(X) \subseteq G_i(X) \cup [G_i(X)]$ and $G_i^t([X]) \subseteq G_i(X) \cup [G_i(X)]$.*

Proof. We will prove this by induction on G_i^t. The statement holds initially for G_1^t by the definition of twin networks. Suppose the statement holds for G_{i-1}^t, i.e. $G_{i-1}^t(X) \subseteq G_{i-1}(X) \cup [G_{i-1}(X)]$ and $G_{i-1}^t([X]) \subseteq G_{i-1}(X) \cup [G_{i-1}(X)]$ for every $\{X, [X]\}$ in G_{i-1}^t, we need to show the statement holds for G_i^t.

For simplicity, let Y denote the variable being eliminated at step i, i.e. $Y = \pi(i-1)$. WLG, consider each base variable X in G_i^t (similar argument can be applied to each duplicate variable $[X]$). $G_i^t(X)$ is affected by the elimination of $\{Y, [Y]\}$ iff X is a neighbor of $\{Y, [Y]\}$ in G_{i-1}^t. Moreover, by induction, X is a neighbor of $\{Y, [Y]\}$ in G_{i-1}^t only if X is a neighbor of Y in G_{i-1}.

When X is not a neighbor of Y, then $G_i^t(X) = G_{i-1}^t(X)$ and $G_i(X) = G_{i-1}(X)$, so the statement holds.

When X is a neighbor of Y, $G_i(X) = G_{i-1}(X) \cup G_{i-1}(Y) \setminus \{Y\}$ by the definition of variable elimination. We can then bound $G_i^t(X)$ as follows:

$$\begin{aligned}
G_i^t(X) &\subseteq (G_{i-1}^t(X) \cup G_{i-1}^t(Y) \cup G_{i-1}^t([Y])) \setminus \{Y, [Y]\} \\
&\quad \text{(eliminating } \{Y, [Y]\} \text{ on } G^t) \\
&\subseteq (G_{i-1}(X) \cup [G_{i-1}(X)] \cup G_{i-1}(Y) \cup [G_{i-1}(Y)]) \setminus \{Y, [Y]\} \\
&\quad \text{(by inductive hypothesis)} \\
&= (G_{i-1}(X) \cup G_{i-1}(Y)) \cup ([G_{i-1}(X)] \cup [G_{i-1}(Y)]) \setminus \{Y, [Y]\} \\
&= G_i(X) \cup [G_i(X)].
\end{aligned}$$

□

Proof for Theorem 1. Consider each variable X that is eliminated at step i, i.e. $X = \pi(i)$. By Lemma 1, $\mathbf{C}^t(X) = G_{i-1}^t(X) \subseteq G_{i-1}(X) \cup [G_{i-1}(X)] = \mathbf{C}(X) \cup [\mathbf{C}(X)]$.

We next bound $\mathbf{C}^t([X])$ if X is an internal variable,

$$\begin{aligned}
\mathbf{C}^t([X]) &\subseteq (G_{i-1}^t([X]) \cup \mathbf{C}^t(X)) \setminus \{X\} \\
&\quad \text{(eliminating } X \text{ from } G_{i-1}^t) \\
&\subseteq G_{i-1}(X) \cup [G_{i-1}(X)] \setminus \{X\} \\
&\quad \text{(by Lemma 1)} \\
&\subseteq \mathbf{C}(X) \cup [\mathbf{C}(X)].
\end{aligned}$$

□

Proof for Corollary 1. By Theorem 1, for every base variable $X \in G^t$, $|\mathbf{C}^t(X)| \leq |\mathbf{C}(X) \cup [\mathbf{C}(X)]| \leq 2|\mathbf{C}(X)|$. Similarly, for every duplicate variable $[X] \in G^t$, $|\mathbf{C}^t([X])| \leq |\mathbf{C}(X) \cup [\mathbf{C}(X)]| \leq 2|\mathbf{C}(X)|$. So $w^t = \max_{X \in G^t} |\mathbf{C}^t(X)| - 1 \leq 2 \max_{X \in G} |\mathbf{C}(X)| - 1 = 2(w+1) - 1 = 2w + 1$. □

Proof for Corollary 2. Consider an optimal elimination order π for base network G with width w. By Corollary 1, the twin elimination order for G^t with width no more than $2w + 1$. It follows that the treewidth of G^t is no more than $2w + 1$. □

Proof of Theorem 2

We first state a key observation from Algorithm 1, which is formulated as Lemma 2. For simplicity, we say a leaf node *hosts variable* X if it hosts a family that contains X.

Lemma 2. *Consider each invariant edge (i, j) in a twin jointree constructed from Algorithm 1. A variable X is hosted in some leaf on the i-side of the edge iff $[X]$ is also hosted in some leaf on the i-side of the edge.*

Proof. If X is a root variable, then $X = [X]$ and the lemma holds. Now suppose X is an internal variable. Let k be the leaf node on the i-side of the edge that hosts X, it suffices to show that k is duplicated into a $[k]$ on the i-side that hosts $[X]$.

Since X is an internal variable, k hosts either the family for X or a family for some child of X. In either case, k hosts a family for an internal variable. Moreover, k is contained in some subtree rooted at u whose leaves host only families for internal variables. Since k is on the i-side of the invariant edge (i, j), u is also on the i-side of the edge. Hence the duplicate leaf $[k]$, which hosts $[X]$, is also on the i-side of the edge.

Conversely, suppose $[X]$ is hosted by some leaf $[k]$ on the i-side of edge (i, j), then $[k]$ hosts some family for a duplicate variable. $[k]$ is contained in some duplicate subtree rooted at $[u]$ whose leaves only host families for duplicate variables. It follows that the base subtree rooted at u is located on the i-side of the edge and contains a base node k hosting X. □

We first recall the definition of separators: $X \in \mathbf{S}_{ij}$ if and only if X is hosted on both sides of the edge (i, j). For simplicity, we use $\texttt{vars}(i, j)$ to denote the variables that appear on the i-side of the edge (i, j) in the base jointree. Similarly, we use $\texttt{vars}^t(i, j)$ and to denote the variables that appear on the i-side in the twin jointree. By definition, for each edge (i, j), $\mathbf{S}_{ij} = \texttt{vars}(i, j) \cap \texttt{vars}(j, i)$ and $\mathbf{S}^t_{ij} = \texttt{vars}^t(i, j) \cap \texttt{vars}^t(j, i)$. Given a jointree and its root, we say that a jointree node j is *above* a jointree node i if j is closer to the root than i, and that j is *below* i if j is further from the root than i.

Proof for Theorem 2. We derive the separators for each type of edges. WLG, for each edge (i, j), assume that j is above i. First consider each duplicated edge (i, j), we have $\texttt{vars}^t(i, j) = \texttt{vars}(i, j)$ by Algorithm 1. Moreover, $\texttt{vars}^t(j, i)$ can only contain extra duplicate variables comparing to $\texttt{vars}(j, i)$. Thus, $\mathbf{S}^t_{ij} = \mathbf{S}_{ij}$.

For each duplicate edge $([i], [j])$, we have $\texttt{vars}^t([i], [j]) = [\texttt{vars}(i, j)]$ by Algorithm 1. We next show that for each $[X] \in \texttt{vars}^t([i], [j])$, $[X] \in \texttt{vars}^t([j], [i])$ iff $X \in \texttt{vars}(j, i)$, which then concludes $\mathbf{S}^t_{[i][j]} = [\mathbf{S}_{ij}]$. We first show the if-part. Let u be the least common ancestor of j and $[j]$. If $X \in \texttt{vars}(j, i)$, then X is hosted by some leaf k that appears either below u, or above u, in the base jointree. Suppose k is below u, then k's duplicate $[k]$ hosts $[X]$ on the $[j]$-side in the twin jointree by Algorithm 1, which implies $[X] \in \texttt{vars}^t([j], [i])$. Suppose k is above u. If u is the root of the jointree, then X is a root variable and $[X] \in \texttt{vars}^t([j], [i])$. Otherwise, let p be the parent of u, then (u, p) is an invariant edge, and $[X]$ appears on the p-side of the edge by Lemma 2.

Similar argument applies for the only-if part. Suppose $[X] \in \texttt{vars}^t([i], [j])$, then $[X]$ is hosted by some duplicate leaf $[k]$ either below or above u. If $[k]$ is below u, then the duplicated node k is also below u. If $[k]$ is above u, then the duplicated node k must also appear above u by Lemma 2.

For each invariant edge (i, j), it follows from Lemma 2 that $\texttt{vars}^t(i, j) = \texttt{vars}(i, j) \cup [\texttt{vars}(i, j)]$ and $\texttt{vars}^t(j, i) = \texttt{vars}(j, i) \cup [\texttt{vars}(j, i)]$. Hence, $\mathbf{S}^t_{ij} = \mathbf{S}_{ij} \cup [\mathbf{S}_{ij}]$. □

Proof for Corollary 3. Let T be a jointree for G, and let T^t be the twin jointree for G^t obtained from Algorithm 1. Let i be a node in jointree T. By Theorem 3, for all neighbors j of i, $\mathbf{S}^t_{ij} \subseteq \mathbf{S}_{ij} \cup [\mathbf{S}_{ij}]$. So $\mathbf{C}^t_i = \bigcup_j \mathbf{S}^t_{ij} \subseteq \bigcup_j \mathbf{S}_{ij} \cup [\mathbf{S}_{ij}] = \mathbf{C}_i \cup [\mathbf{C}_i]$. So $|\mathbf{C}^t_i| \leq |\mathbf{C}_i \cup [\mathbf{C}_i]| \leq 2|\mathbf{C}_i|$. So $w^t = \max_{i \in T^t} |\mathbf{C}^t_i| - 1 \leq 2 \max_{i \in T} |\mathbf{C}_i| - 1 = 2(w + 1) - 1 = 2w + 1$.

The number of nodes in the twin jointree is at most twice the number of nodes in the base jointree since Algorithm 1 adds at most one duplicate for each node in the base jointree. □

Proof of Theorem 3

Proof for Theorem 3. From [5], a thinned jointree can be obtained by applying a sequence of thinning rules to a base jointree. Let $\mathbf{Q} = \{Q_1, \ldots, Q_T\}$ denote the T thinning rules being applied to the base jointree in order. We next construct a thinning sequence $\mathbf{Q}^t$ for the twin jointree, which leads to the thinned separators defined in the Theorem.

We first define a *parallel thinning step* as simultaneously thinning a functional variable X from the base jointree and thinning $\{X, [X]\}$ from the twin jointree. Consider any thinning rule Q_i that thins X from a separator $\mathbf{S}_{ij}$, the parallel thinning on $\mathbf{S}^t$ is defined as follows:

- Suppose (i, j) is a duplicated edge, then we thin X from $\mathbf{S}^t_{ij}$ and $[X]$ from $\mathbf{S}^t_{[i][j]}$
- Suppose (i, j) is an invariant edge, then we thin $\{X, [X]\}$ from $\mathbf{S}^t_{ij}$

First note that the definition of parallel thinning ensures that the relation between $\mathbf{S}$ and $\mathbf{S}^t$ specified in the theorem holds after every parallel thinning step. It remains to show that the parallel thinnings on $\mathbf{S}^t$ are indeed valid.

Let $\mathbf{S}$, $\mathbf{S}^t$ denote the separators for the base and twin jointree during the parallel thinnings. Consider a parallel thinning step being applied to a duplicated edge, which, by definition, removes X from $\mathbf{S}_{ij}$, X from $\mathbf{S}^t_{ij}$, and $[X]$ from $\mathbf{S}^t_{[i][j]}$. Suppose the removal of X from $\mathbf{S}_{ij}$ is supported by the first thinning rule, i.e. the edge (i, j) is on the path between two leaf nodes, call them l and r, both

hosting the family of X, and every separator on that path contains X. We claim that the removal of X from $\mathbf{S}^t_{ij}$ and the removal of $[X]$ from $\mathbf{S}^t_{[i][j]}$ are both supported by the first thinning rule. By Algorithm 1, the leaf nodes $\{l, r\}$ host the family of X, and the leaf nodes $\{[l], [r]\}$ host the family of $[X]$ in the twin jointree. Moreover, by the inductive assumption on separators, X appears on every separator between l and r and $[X]$ appears on every separator between $[l]$ and $[r]$. This is based on an observation that the path from l to r can be divided into three sub-paths: a sub-path consisting of only duplicated edges ($l = p_1, \ldots, p_s$), a sub-path consisting of only invariant edges ($p_s, \ldots, p_m$), and a sub-path consisting of only duplicated edges ($p_m, \ldots, p_n = r$), where $1 < s \leq m < n$[6]. Given the path from l to r, we can then express the path from $[l]$ to $[r]$ as three sub-paths as well: a sub-path consisting of only duplicate edges ($[l] = [p_1], \ldots, p_s$), a sub-path consisting of only invariant edges ($p_s, \ldots, p_m$), and a sub-path consisting of only duplicate edges ($p_m, \ldots, [p_n] = [r]$). For each duplicated edge (i, j) on the sub-path from l to p_s or on the sub-path from p_m to r, $X \in \mathbf{S}_{ij}$ iff $[X] \in \mathbf{S}_{[i][j]}$. For each invariant edge (i, j) on the sub-path from p_s to p_m, $X \in \mathbf{S}_{ij}$ iff $[X] \in \mathbf{S}_{ij}$.

Suppose the removal of X from $\mathbf{S}_{ij}$ is supported by the second thinning rule, i.e. no other separators $\mathbf{S}_{ik}$ contains X, or no other separators $\mathbf{S}_{kj}$ contains X. Then the removal of X from $\mathbf{S}^t_{ij}$ and the removal of $[X]$ from $\mathbf{S}^t_{[i][j]}$ are both supported by the second thinning rule due to the inductive assumption on separators.

Consider now a parallel thinning step being applied to an invariant edge which removes X from $\mathbf{S}_{ij}$ and $\{X, [X]\}$ from $\mathbf{S}^t_{ij}$. Suppose the removal of X from $\mathbf{S}_{ij}$ is supported by the first thinning rule, where the edge (i, j) is on the path between two leaf nodes l and r both hosting the family of X and every separator on that path contains X. Again, by the inductive assumption on separators, $[X]$ appears in all separators on the path between $[l]$ and $[r]$, which also includes the invariant edge (i, j). Hence, $[X]$ can also be removed from $\mathbf{S}^t_{ij}$ using the first thinning rule. Suppose the removal of X from $\mathbf{S}_{ij}$ is supported by the second thinning rule, then we can apply the second thinning rule to remove $\{X, [X]\}$ from $\mathbf{S}^t_{ij}$ due to the inductive assumption on separators. □

Proof of Theorem 4

We first extend the notion of twin graph sequence to *N-twin graph sequence*, denoted as $G^N_1, G^N_2, \ldots, G^N_n$, where G^N_1 is the moral graph for G^N, and G^N_{i+1} is the result of eliminating $\{\pi(i)^1, \ldots, \pi(i)^N\}$ from G^N_i. For each variable X^j ($j \in \{1, \ldots, N\}$) in G^N_i, let $G^N_i(X^j)$ be the set consisting of X^j and its neighbors in G^N_i. For a set of variables $\mathbf{X}$, we use $\mathbf{X}^j$ to denote $\{X^j | X \in \mathbf{X}\}$.

Lemma 3. *Consider a base network G and its N-twin network G^N. Let π be an elimination order for G. Then for each variable X^j ($j \in \{1, \ldots, N\}$) in G^N_i, $G^N_i(X^j) \subseteq \bigcup_{k=1}^N G_i(X)^k$.*

Proof. We will prove this by induction on G^N_i. The statement holds initially for G^N_1 by the definition of N-twin networks. Suppose the statement holds for G^N_{i-1}, i.e. $G^N_{i-1}(X^j) \subseteq \bigcup_{k=1}^N G_{i-1}(X)^k$ for every variable $X^j \in G^N_{i-1}$, we need to show the statement holds for G^N_i.

Let $Y = \pi(i-1)$. Suppose we eliminate variables $\{Y^k\}_{k=1}^N$ from G^N. $G^N_i(X)$ is affected by the elimination of $\{Y^k\}_{k=1}^N$ iff X is a neighbor of $\{Y^k\}_{k=1}^N$ in G^N_i. Moreover, by induction, X is a neighbor of $\{Y^k\}_{k=1}^N$ in G^N_i only if X is a neighbor of Y in G_i.

When X is not a neighbor of Y, then $G^N_i(X^j) = G^N_{i-1}(X^j)$ for all $j = 1, 2, \ldots, N$ and $G_i(X) = G_{i-1}(X)$, so the statement holds.

[6]Note that we do not preclude the possibility of having no invariant edge on the path.

When X is a neighbor of Y, for all $j = 1, 2, \ldots, N$, we can then bound $G_i^N(X^j)$ as follows:

$$\begin{aligned} G_i^N(X^j) &\subseteq G_{i-1}^N(X^j) \cup \bigcup_{k=1}^{N} (G_{i-1}^N(Y^k)) \setminus \{Y^k\}_{k=1}^N \\ &\subseteq (\bigcup_{k=1}^{N} G_{i-1}(X)^k) \cup (\bigcup_{k=1}^{N} G_{i-1}(Y)^k) \setminus \{Y^k\}_{k=1}^N \\ &\quad \text{(by induction hypothesis)} \\ &= \bigcup_{k=1}^{N} (G_{i-1}(X)^k \cup G_{i-1}(Y)^k \setminus \{Y^k\}) \\ &= \bigcup_{k=1}^{N} G_i(X)^k. \end{aligned}$$

□

Lemma 4. *Let G^N be an N-twin network and let $j \in \{1, \ldots, N\}$ be a positive integer. Then $\mathbf{C}^N(X^j) \subseteq \bigcup_{k=1}^{N} \mathbf{C}(X)^k$.*

Proof. Let $X = \pi(i)$. By Lemma 3, $\mathbf{C}^N(X^1) = G_i^N(X^1) \subseteq \bigcup_{k=1}^{N} G_i(X)^k = \bigcup_{k=1}^{N} \mathbf{C}(X)^k$. We next bound $\mathbf{C}^N(X^j)$ where $j \in \{2, \ldots, N\}$ by induction. For each $j \in \{2, \ldots, N\}$, assume $\mathbf{C}^N(X^h) \subseteq \bigcup_{k=1}^{N} \mathbf{C}(X)^k$ for $h = 1, 2, \ldots, j-1$, then

$$\begin{aligned} \mathbf{C}^N(X^j) &\subseteq G_i^N(X^j) \cup \bigcup_{k=1}^{j-1} \mathbf{C}^N(X^k) \quad \text{(by VE definition)} \\ &\subseteq (\bigcup_{k=1}^{N} G_i(X)^k) \cup (\bigcup_{k=1}^{N} \mathbf{C}(X)^k) \quad \text{(by inductive hypothesis)} \\ &\subseteq \bigcup_{k=1}^{N} \mathbf{C}(X)^k \end{aligned}$$

□

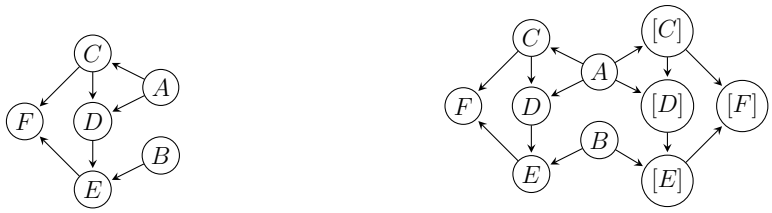
Proof for Corollary 4. By Lemma 4, for all $X^j \in G^N$, $|\mathbf{C}(X^j)| \leq \sum_{k=1}^{N} |\mathbf{C}(X)^k| = N|\mathbf{C}(X)|$. Therefore, the width w^N of π^N is

$$\begin{aligned} w^t &= \max_X \mathbf{C}^N(X) - 1 \\ &\leq N \max_X \mathbf{C}(X) - 1 \\ &= N(w+1) - 1 \end{aligned}$$

For causal treewidth, we can extend Algorithm 1 to construct N-twin jointrees by making $N-1$ duplicates for each duplicated subtree, and construct N-twin thinned jointrees by extending Theorem 3. By analogous arguments, we can show that $w^t = N(w+1) - 1$. □

B Tightness of Bounds

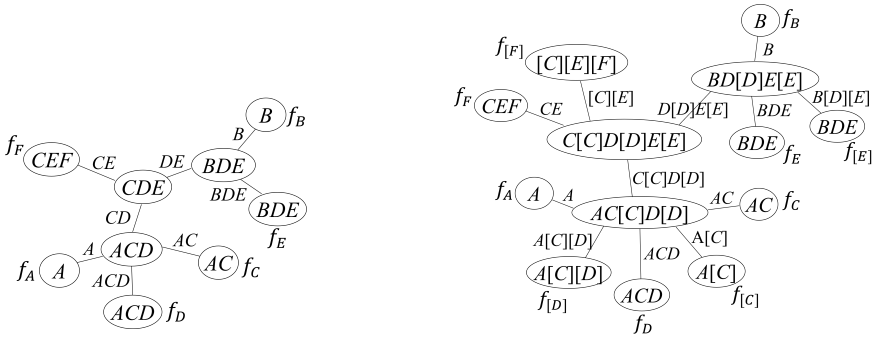
As we claimed in the paper, Corollaries 1 and 3 provide tight bounds for the widths of twin elimination orders defined by Definition 2, and twin jointrees constructed by Algorithm 1, respectively. Moreover, we provide a concrete example where a base network has treewidth w and its twin network has treewidth $2w$, which suggests that our bound on treewidth in Corollary 2 may also be tight (we found these examples through exhaustive search which we were able to do on small examples only, given the allotted computational resources).



(a) base network, width of π is $w = 2$

(b) twin network, width of π^t is $w^t = 5$

Figure 7: The elimination order π is (A, B, F, D, C, E) and the twin elimination order π^t is $(A, B, F, [F], D, [D], C, [C], E, [E])$. We have $w^t = 2w + 1$.



(a) base jointree, width $w = 2$

(b) twin jointree, width $w^t = 5$

Figure 8: A base jointree and its twin jointree, where $w^t = 2w + 1$.

For the tightness of Corollary 1, consider the base network G shown in Figure 7a. The elimination order $\pi = (A, B, F, D, C, E)$ has width of 2. The twin network G^t for G is shown in Figure 7b. By Definition 2, the twin elimination order for G^t is $\pi^t = (A, B, F, [F], D, [D], C, [C], E, [E])$ which has a width of 5.

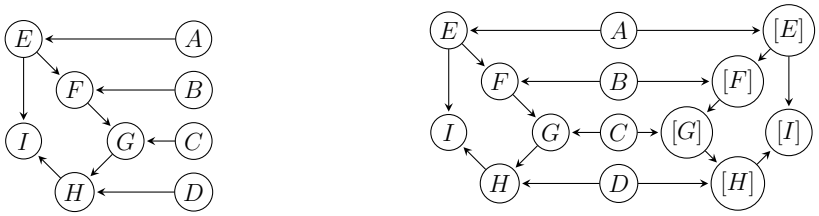
For the tightness of Corollary 3, Figure 8a shows a base jointree for the base network G. This base jointree was constructed using elimination order π and it has a width of 2. Figure 8b shows a twin jointree constructed by Algorithm 1 which has a width of 5.

For our last example, Figure 9a shows a base network with a treewidth of 2. Figure 9b shows the corresponding twin network which has a treewidth of 4. This shows that if our treewidth bound in Corollary 2 is not tight, then it is off by at most 1.

C More on Experiments

Our experiments were run on 2.60GHz Intel Xeon E5-2670 CPU with 256 GB of memory.

In the main paper, we plotted the jointree widths for random networks with a maximum number of parents ($p = 5$). Here we show the complete experimental results for the random networks with



(a) base network, treewidth $w = 2$

(b) twin network, treewidth $w^t = 4$

Figure 9: A base network and its corresponding twin network, where $w^t = 2w$.

num vars	max pars	BASE-MF		TWIN-ALG1		TWIN-MF		BASE-MF-RLS		TWIN-THM3		TWIN-MF-RLS	
		wd	nwd	wd	nwd	wd	nwd	wd	nwd	wd	nwd	wd	nwd
50	3	7.5	11.5	14.4	17.1	10.0	14.0	5.2	11.9	7.6	13.2	5.6	13.0
	5	14.3	17.4	26.8	29.2	15.9	19.6	7.2	15.0	10.0	16.2	7.3	16.0
	7	19.5	22.4	36.9	39.1	20.4	24.1	8.9	17.1	12.2	18.3	8.9	18.1
75	3	10.2	13.8	18.9	21.5	14.5	17.9	6.1	12.9	8.6	14.3	6.9	14.0
	5	21.3	23.9	39.1	41.1	23.9	27.2	8.4	16.4	11.9	17.8	9.1	17.5
	7	28.8	31.4	54.0	56.0	30.3	33.7	11.0	18.6	16.4	21.2	11.7	19.7
100	3	13.7	16.7	25.1	27.5	19.9	22.8	7.1	13.8	10.2	15.3	8.3	14.9
	5	27.1	29.7	50.8	52.9	31.4	34.5	9.5	17.1	13.5	18.9	11.2	18.4
	7	38.7	40.9	72.9	74.7	40.8	43.8	14.5	20.3	22.9	26.6	15.9	21.9
125	3	16.7	19.6	30.8	33.0	26.9	29.3	7.9	14.6	10.5	16.1	9.2	15.9
	5	34.9	37.3	65.1	66.9	39.1	42.1	12.1	18.3	17.2	21.4	14.8	20.3
	7	47.9	50.1	89.8	91.7	50.6	53.5	19.7	23.7	32.5	35.1	21.7	26.1
150	3	20.3	22.9	36.9	38.9	31.8	34.2	8.6	15.2	11.1	16.7	9.8	16.4
	5	41.4	43.6	76.9	78.6	49.1	51.7	14.6	19.7	21.2	24.7	18.6	22.8
	7	56.7	58.7	106.7	108.2	60.6	63.3	22.5	26.1	36.8	39.3	25.6	29.4
200	3	25.8	28.2	47.2	49.1	42.9	44.8	10.7	16.4	13.6	18.3	12.3	17.9
	5	55.1	57.2	102.4	104.0	65.7	68.0	19.9	23.8	29.5	32.3	26.0	29.4
	7	75.3	77.4	141.5	142.1	80.4	82.7	34.3	37.1	57.5	59.8	39.0	42.3
250	3	31.6	34.0	58.1	60.0	55.7	57.4	12.1	17.4	15.0	19.5	14.1	19.2
	5	68.6	70.5	128.5	129.9	84.9	86.8	25.8	28.9	38.9	41.3	36.1	38.9
	7	94.3	96.0	178.7	180.0	102.0	104.0	44.1	46.7	75.9	77.9	50.5	53.7
300	3	40.1	42.3	73.9	75.6	69.9	71.5	13.9	18.4	17.4	21.2	16.3	20.5
	5	82.4	84.1	153.3	154.5	104.8	106.4	32.8	35.7	50.1	52.3	46.1	48.6
	7	115.3	116.9	218.5	219.8	124.5	126.3	55.9	58.5	99.1	100.9	65.7	68.7

Table 1: Widths (wd) and normalized widths (nwd) of various twin jointrees under rNET [8]. Refer to the main paper for the details of different jointree construction methods. All the thinned jointrees are constructed by bounding the functional chain lengths by 10; see [5] for details.

$p = 3, 5, 7$ generated according to the method of [8] which we discussed in the main paper. We record the widths and normalized widths for each twin jointrees. Recall that if a jointree has clusters $\mathbf{C}_1, \ldots, \mathbf{C}_n$, then the normalized width is $\log_2 \sum_{i=1}^{n} 2^{|\mathbf{C}_i|}$. Table 1 shows the complete results for rNET and Table 2 shows the complete results for rSCM.

In addition, we report results on random networks generated by a second method proposed in [15]. Given a number of nodes n and a maximum degree[7] d for each node, the method generates a random network by repeatedly adding/removing random edges from the current DAG. We refer to these random networks as rNET2. Similarly to what we did for rNET, we then added a unique root as parent for each internal variable in rNET2. We refer to these modified networks as rSCM2. For each combination of $n \in \{50, 100, 150, 200\}$ and $d \in \{5, 10, 15\}$, we generated 50 random base networks and reported the average widths and normalized widths for each twin jointree. Table 3 shows the complete results for rNET2 and Table 4 shows the complete results for rSCM2. The patterns in these tables resemble the ones for rNET and rSCM.

[7]The degree of a node is the number of its parents and children.

num vars	max pars	BASE-MF		TWIN-ALG1		TWIN-MF		BASE-MF-RLS		TWIN-THM3		TWIN-MF-RLS	
		wd	nwd	wd	nwd	wd	nwd	wd	nwd	wd	nwd	wd	nwd
50	3	7.5	11.9	14.4	17.4	14.0	17.1	11.1	15.6	14.7	18.3	14.2	18.3
	5	14.3	17.4	26.9	29.3	26.4	28.9	19.0	22.7	23.5	26.5	22.9	26.2
	7	19.5	22.4	37.0	39.3	37.0	39.1	24.3	27.7	31.0	33.5	29.4	32.4
75	3	10.2	14.0	18.9	21.6	18.5	21.3	14.9	18.5	19.1	22.1	18.1	21.5
	5	21.3	23.9	39.1	41.1	38.5	40.7	26.9	29.7	33.5	35.8	31.8	34.5
	7	28.8	31.4	54.1	56.2	53.9	55.9	34.5	36.9	44.9	46.9	42.3	44.5
100	3	13.7	16.8	25.1	27.5	24.4	26.9	19.5	22.4	24.9	27.1	23.1	25.9
	5	27.1	29.7	50.8	52.9	50.8	52.8	34.6	37.0	43.9	45.8	40.5	42.8
	7	38.7	40.9	73.0	74.8	72.8	74.5	45.8	47.8	61.3	62.9	57.4	59.4
125	3	16.7	19.6	30.8	33.0	30.5	32.7	23.8	26.5	29.5	31.8	28.0	30.6
	5	34.9	37.3	65.1	66.9	64.9	66.6	44.4	46.5	56.5	58.1	51.8	54.0
	7	47.9	50.1	90.1	91.9	89.9	91.6	55.4	57.5	76.1	77.6	71.2	73.2
150	3	20.3	22.9	36.9	38.9	36.2	38.2	27.3	30.0	34.0	36.1	31.9	34.3
	5	40.4	43.6	76.9	78.6	76.3	77.9	50.6	52.5	65.3	66.8	60.6	62.5
	7	56.7	58.7	106.7	108.2	106.0	107.5	65.7	67.4	91.1	92.4	83.2	85.0
200	3	25.8	28.3	47.2	49.1	46.2	48.2	35.5	37.6	44.2	45.9	40.7	42.8
	5	55.0	57.2	102.5	104.1	102.0	103.5	67.7	69.4	88.7	90.0	82.6	84.1
	7	75.3	77.4	141.5	143.1	75.3	77.4	87.0	88.6	124.6	125.8	115.2	116.7
250	3	31.6	34.0	58.1	60.0	57.5	59.4	43.8	45.8	54.3	55.9	49.8	51.9
	5	68.6	70.5	128.5	129.9	127.1	128.5	82.1	83.6	110.3	111.5	101.7	103.2
	7	94.3	96.0	178.7	180.0	178.3	179.7	108.5	109.9	158.9	160.0	146.8	148.1
300	3	40.1	42.3	73.9	75.6	72.4	74.1	54.1	55.8	66.7	68.1	61.8	63.6
	5	82.5	84.1	153.3	154.5	152.7	154.0	99.1	100.4	135.0	136.2	124.3	125.8
	7	115.3	116.9	218.5	219.8	206.8	208.1	131.2	132.5	194.4	195.6	180.1	181.4

Table 2: Widths (wd) and normalized widths (nwd) of various twin jointrees under rSCM [8]. Refer to the main paper for the details of different jointree construction methods. All the thinned jointrees are constructed by bounding the functional chain lengths by 10; see [5] for details.

num vars	max pars	BASE-MF		TWIN-ALG1		TWIN-MF		BASE-MF-RLS		TWIN-THM3		TWIN-MF-RLS	
		wd	nwd	wd	nwd	wd	nwd	wd	nwd	wd	nwd	wd	nwd
50	5	20.0	22.4	37.3	39.0	20.3	23.6	8.7	14.5	13.5	17.1	8.7	15.5
	10	32.9	35.4	55.5	57.7	32.9	36.4	13.7	18.6	21.6	24.7	13.7	19.6
	15	38.1	40.7	62.5	64.9	38.1	41.7	19.7	23.4	30.6	33.1	19.7	24.4
100	5	40.0	41.8	76.7	78.1	40.1	42.7	14.5	18.4	23.5	25.8	14.6	19.4
	10	65.1	67.4	110.6	112.7	65.5	68.6	33.0	35.9	52.3	54.5	33.0	36.9
	15	75.1	77.5	114.7	117.1	75.2	78.5	45.3	48.0	66.0	68.5	45.3	49.0
150	5	59.7	61.2	110.6	112.0	60.3	62.4	21.5	24.5	36.6	38.6	21.5	25.5
	10	97.1	99.0	159.6	161.7	97.3	100.1	49.8	52.5	80.9	83.1	49.8	53.5
	15	109.7	111.8	168.9	171.3	109.8	112.8	66.7	69.1	92.7	95.3	72.3	75.6
200	3	109.7	111.8	168.9	171.3	109.8	112.8	66.7	69.1	92.7	95.3	72.3	75.6
	5	127.0	128.5	193.6	195.8	127.2	129.6	65.0	67.4	97.2	99.6	65.0	68.4
	7	141.1	142.7	210.8	212.9	141.3	143.9	82.9	85.2	121.3	123.7	82.9	86.3

Table 3: Widths (wd) and normalized widths (nwd) of various twin jointrees under rNET2 [15]. Refer to the main paper for the details of different jointree construction methods. All the thinned jointrees are constructed by bounding the functional chain lengths by 10; see [5] for details.

num vars	max pars	BASE-MF		TWIN-ALG1		TWIN-MF		BASE-MF-RLS		TWIN-THM3		TWIN-MF-RLS	
		wd	nwd	wd	nwd	wd	nwd	wd	nwd	wd	nwd	wd	nwd
50	5	20.0	22.4	36.2	39.9	37.2	39.2	27.9	30.2	39.6	41.3	37.1	39.1
	10	32.9	35.4	64.8	66.7	64.9	66.8	37.4	39.8	54.0	55.8	50.6	52.9
	15	38.1	40.7	75.3	77.4	76.4	78.5	40.8	43.4	61.1	63.2	60.3	62.5
100	5	40.1	41.7	78.4	79.8	77.7	79.1	54.8	56.2	80.6	81.8	74.3	75.7
	10	65.1	67.4	129.3	131.0	128.6	130.4	75.1	76.6	116.2	117.5	106.4	108.2
	15	75.1	77.5	149.6	151.5	149.3	151.2	81.2	83.1	131.0	132.5	125.7	127.8
150	5	59.7	61.1	118.5	119.7	118.1	119.4	83.0	84.3	125.1	126.2	113.8	115.0
	10	97.1	99.0	193.4	194.9	193.3	194.8	111.1	112.5	174.7	175.8	161.1	162.7
	15	109.7	91.8	218.8	220.4	218.8	220.4	120.3	121.9	196.6	197.9	186.0	187.7
200	5	75.6	76.6	150.7	151.8	150.8	151.9	102.5	103.6	155.0	156.1	143.3	144.4
	10	127.0	128.5	253.4	254.6	252.7	254.0	149.4	148.6	233.0	234.2	212.9	214.3
	15	141.1	142.7	281.7	283.0	281.5	282.8	156.3	159.6	255.6	256.7	238.6	240.2

Table 4: Widths (wd) and normalized widths (nwd) of various twin jointrees under rSCM2 [15]. Refer to the main paper for the details of different jointree construction methods. All the thinned jointrees are constructed by bounding the functional chain lengths by 10; see [5] for details.