Graph Comparison for Causal Discovery

Joseph Cottam, Maria Glenski, Zhuanyi Huang Shaw,
Ryan Rabello, Austin Golding, Svitlana Volkova, Dustin Arendt

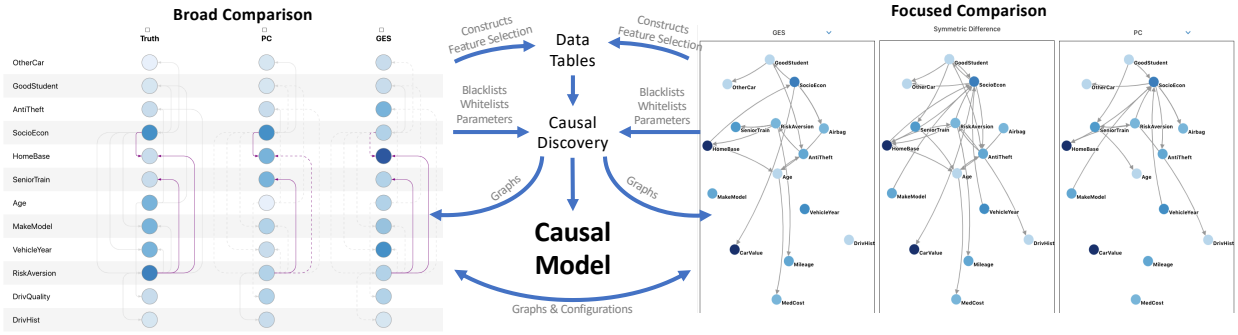


Fig. 1: Graph comparison views and abstract workflow. Data tables are processed through causal discovery algorithms. Focused and broad comparison tools guide algorithm selection, parameters tuning and building black/white lists. Tasks motivate changing between parts of the process. Ultimately an analysts creates a causal model through exploration in a Jupyter notebook.

Abstract—Reasoning about cause and effect is one of the frontiers for modern machine learning. Many causality techniques reason over a "causal graph" provided as input to the problem. When a causal graph cannot be produced from human expertise, "causal discovery" algorithms can be used to generate one from data. Unfortunately, causal discovery algorithms vary wildly in their results due to unrealistic data and modeling assumptions, so the results still need to be manually validated and adjusted. This paper presents a graph comparison tool designed to help analysts curate causal discovery results. This tool facilitates feedback loops whereby an analyst compares proposed graphs from multiple algorithms (or ensembles) and then uses insights from the comparison to refine parameters and inputs to the algorithms. We illustrate different types of comparisons and show how the interplay of causal discovery and graph comparison improves causal discovery.

1 Introduction

Data analysts are called on more and more to explain *why* something happens. Explaining why requires going beyond descriptive methods of *what* has occurred, and looking to underlying mechanisms. This deeper exploration is the realm of causal reasoning [35, 36]. Creating causal models is an interactive process, relying on human intuition and experience to review algorithmic results. Therefore, it is an ideal human-machine teaming opportunity.

Causal discovery algorithms attempt to surface causal relationships from data, these relationships are typically represented as a graph. However, as many datasets are not collected with causal discovery in mind, most causal discovery algorithms can (at best) provide only candidate causal relationships. Furthermore, different algorithms may propose conflicting causal structures based on (1) different assumptions about the underlying data, (2) inherent limitations in the datasets (e.g., not fully observing processes), and (3) randomness used in the algorithms. As a result, it is incumbent on the analyst to select appropriate algorithms and interpret results.

Visual analytics can help with causal discovery in particular by facilitating graph comparison, thereby helping an analyst curate algorithm outputs, combine results, and iterate on algorithm inputs (such as relationship black/white-listing or feature selection). The starting point for this workflow is a set of candidate graphs produced by causal discovery algorithms. The analyst must then select between different causal discovery algorithms (or ensembles of them) and possibly revise the inputs to those algorithms based on their observations and tacit knowledge. Ultimately, the analyst is trying to find a ***causal model*** that captures the variation of the data and conforms to known truths (or construct new models based on compelling evidence).

This paper makes two contributions: (1) We argue that the "graph comparison" task is a necessary part of the causal discovery workflow, e.g., during validation, and furthermore, that causal discovery is a relevant but under-explored application domain in visual analytics. (2) We present a graph comparison tool for causal discovery workflows. While the tool builds on existing graph visualization techniques, we assert that the workflows it supports are novel and effective in the context of causal discovery.

1.1 Motivation

Causal reasoning is essential to advancing modeling beyond simply *describing* data. More and more problems require models for *predicting* future behavior, *prescribing* ways to change systems, or *imagining* alternatives. These latter problems require a model of causal relationships [35, 36]. Algorithmic causal discovery of these relationships is fundamentally limited in many cases, even under ideal circumstances [18, 20]. In practice, it is common for different algorithms to produce incompatible outputs given the same input.

Though fundamentally a machine-learning problem, it is naive to apply causal discovery in a completely automated fashion; models are evaluated not just on their ability to capture variation in the data but also on the feasibility of their *structure*. Much of the relevant knowledge is not exposed in a computationally accessible form. It is held in research papers and the tacit knowledge of researchers. Qualitative evaluation is an essential part of identifying good causal models. Despite its fundamental issues, causal discovery is employed in fields such as economics and epidemiology to build rich models from observational

- *All authors are with Pacific Northwest National Lab.*
- *Emails are first.last@pnnl.gov*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

Tool	*Comments*
GraphViz [8]	Single-graphs, not interactive
Cytoscape [44]	Single-graphs, assumes biology
Cytoscape plugins [13, 28, 31]	Limited graph sizes, few graphs
Phylogenetic trees [33, 40]	Assumes a tree structure
Haase Diagrams [27]	Assumes time, focus on tracing
Animated Geometry [9–11]	Focus on tracing
Regression Explorer [6]	Focus on statistical evidence
DiffAni [41]	Assumes time

Table 1: Tools for graph comparison. Using these tools requires leaving the notebook environment, and thus add workflow complexity.

and interventional data [16]. Observing analysts using causal discovery toolboxes (such as pcalg [23]), we noticed several issues that visual analytics can help address.

In a typical workflow, analysts create models with several different algorithms. Each algorithm produces a ***causal graph*** that represents cause and effect relationships as links between nodes that represent variables (cause *!* effect). (In this paper, all 'causal models' are 'causal graphs', and the terms are used interchangeably). Analysts would render these causal graphs in a generic tool (e.g., as PDFs using graphviz [8]) and perform a side-by-side comparison between the artifacts to identify similarities and differences. This side-by-side comparison stood out as a potential area for improvement as it was manual, slow, and error-prone. Analysts often lost the context for their comparison as they moved between tools and had to write short programs to filter/align graphs as their analyses progressed.

In a survey of existing graph comparison tools (summarized in Table 1), we found that most were focused on specific domains [13, 14, 19, 28, 31, 33, 40, 44], assumed restrictions on the graphs that were not generally true [11, 14], or did not integrate well with the desired workflows [8, 44]. These observations led us to build a new graph comparison-focused tool that can be embedded directly in Jupyter notebooks [25, 38], which allows analysts to conduct their work in a single workspace.

2 Related Work

Graph Comparison In general aligning two graphs is in NP [42]. However, features of the causal discovery domain makes it more tractable. For example, meaningful node labels are often present in causal networks, provided by the input data table's column labels. This makes set-based aggregate comparison metrics, such as the Jaccard index [15, 29], easy to apply and interpret.

There are many biologically focused tools in graph analysis. Cytoscape is a popular application/plugin-framework in the area [44] with plugins specifically for graph comparison (e.g., [13,28,31]). These tools tend to use traditional node-link layouts, and influenced our "focused comparison" view. Phylogentic tree comparison has also benefited from working with simpler graph structures [33, 40]. Though our applications do not contend with the issues of scale, the focus+context, highlighting differences (with or without a canonical reference), and filtering techniques remain applicable.

Dynamic Graph Visualization Graph comparison is related to dynamic graph visualization—in dynamic graph visualization the user compares graphs from different times to understand what has changed. Thus, many lessons learned from dynamic graph visualization can be applied to graph comparison, especially if comparing multiple graphs arranged in some reasonable sequence. The taxonomy presented by Beck *et al.* [2] explores the breadth of the design space in dynamic graph visualization, revealing many options for graph comparison. From this taxonomy, juxtaposed, superimposed, and integrated techniques appear in various aspects of graph comparison tool.

TimeArcTrees [14] was influential in the "broad comparison" view of our tool. TimeArcTrees have two main parts: a hierarchy view and a graph evolution view. The graph evolution shows time as a series of graph snapshots along X axis where the nodes of each snapshot are shown as a single row on the Y axis. Nodes have a fixed Y position across all time slices. The layout is reminiscent of a grid where the nodes are the rows and time points are the columns. Edges are shown as arrows that link between nodes. Edges that go up the Y axis are on the left side of the node icons, while edges that go down the Y axis are on the right. The order of the Y axis is derived from the node hierarchy, with edge crossings minimized within a hierarchy level. This layout makes collapsing elements of the hierarchy as simple as reducing all nodes in a span of the Y axis in to a single summary node. Adjacent temporal slices can be combined in a similar way along the X axis. Our application does not include temporal ordering (for the horizontal order) or hierarchical elements (for the vertical order), limiting the ability of the tool to summarize. However, the organization principles are still useful and our externally created ensembles are conceptually similar to TimeArcTree summaries. Many techniques in the "focused comparison" tool are related to those found in GraphDiaries [1]. In particular, modifying edge and node weights on graph diagrams that share a layout to highlight similarities/differences can be found in GraphDiaries (among other places). The task of this paper can mapped into the taxonomy of GraphDiaries, though it is difficult to directly apply because the analog of 'time' is discrete and branching. Furthermore, we avoid animation in most cases because there are fewer graphs to represent.

Causal Visualization Prior work on causal discovery has focused more on the mechanics of causal discovery, e.g., how the underlying observational data and causal relationships are related. For example, the Visual Causality Analyst tool (VCA) [46] presents the results of causal discovery but much of the focus is on showing the underlying evidence for the graph. This includes prominently featuring tables with statistical summaries and integrating views for examining individual records in a dataset. Visualizations of conditional independence and interactive regression exploration [6] are similarly focused on low-level mechanisms, rather than overall model comparison. The causal diagrams presented show many important parts of causal models and illustrate the types of data encodings that are needed to understand causal relationships. Wang and Mueller's follow-up work on VCA looks at multiple models, but the comparison is again focused on the mechanics that built the model or comparison of the results the models imply [47]. The work in this paper is complementary, comparing structures to elicit decisions about the input to a simulation.

Other tools approach causality more abstractly using animation [9–11] and focus on reasoning about the model's construction. Rather than asking questions about 'what if the model were different', these papers focus on 'what does the model say'. A major contribution of this prior work is studies that indicate that directly representing time (e.g., in Hasse diagrams [27]) is not essential to understanding causal models.

Causal Reasoning & Discovery The graphs discussed in this paper are the results of causal discovery algorithms. A graph representation is essential to modern notions of causal reasoning [35, 36], and often restricted to a directed acyclic graph (DAG). The nodes of the causal graph represent the variables of the causal model. The links are directed and represent proximate causes, where information flows from one variable to the next in the direction of the arrow. This model captures both direct and indirect causal relationships through graph traversals. Recent advances in causal reasoning enable reasoning on non-DAG graphs (cyclic or not fully directed), but reasoning is more limited [20]. A DAG remains the most common representation.

Causal discovery algorithms attempt to build causal graphs from tabular data . There are many algorithms [12, 22], though there are fundamental limitations to what they can discover [49]. Several algorithms are available through statistical packages such as pcalg [18, 23]. This paper uses the GS (Grow-Shrink) [30], PC (Peter-Clark) [5], MMPC (Max-Min Parents-Children) [45], and GES (Greedy Equivalence Search) [4] algorithms, which are readily available in R (pcalg) or python (causal discovery toolkit/CDT [21]). These are common benchmark algorithms (e.g., [50]). Many techniques for causal discovery overlap with Bayesian networks modeling (e.g., the BNLearn library [43] is a dependency for CDT).

DAGs are not not the only causal models. Given the fundamental limitations of causal discovery, the output of causal discovery algorithms

is not always a DAG. Partial ancestral graphs (PAGs) or maximal ancestral graphs (MAGs) represent families of DAGs and honestly report ambiguity (e.g., rather than randomly reporting a specific DAG) [39,48]. An ensemble of individual causal models also may not be a DAG (even if all inputs were DAGs). For the purposes of this paper, all such graphs will be considered 'causal models'.

3 Graph Comparison for Causal Discovery

Graph comparison is a natural part of causal discovery because it enables exploration of many causal discovery results. By restricting the problem domain to causal discovery, many of the complex and intractable aspects of graph comparison are reduced. The overall goal in causal discovery is to find a suitable model of a dataset. What makes a model "best" depends on the problem that the causal model is addressing. Because there are application-specific fitness metrics and limited ability to provide expert knowledge to the discovery algorithms, the overall causal discovery process benefits from interactive tools that enable cycles of feedback or refinement to algorithm inputs.

Algorithmic approaches to causal discovery are inherently limited [20,48]. In most cases, the best that can be provided is a Markov Equivalence Class, where not all edges are directed [39]. To be tractable, the algorithms make simplifying assumptions about either the structure of the causal graph or the properties of the data that may not be testable. Therefore, using causal discovery algorithms should be treated as *exploring* possible structures suggested by the algorithms (rather than identifying a precise model). Graph comparison is a natural part of that exploration. For causal discovery, this is a (mostly) *labeled* graph comparison, significantly simplifying the problem.

We worked closely with a team of analysts using causal discovery algorithms to identify the questions they were trying to answer about causal discovery. The questions included "Where do these graphs agree?", "Where do these graphs differ?", and "How does this graph compare to my understanding of the underlying process?" Answering these questions involved an iterative investigation of inspecting results and modifying algorithm parameters.

Tasks the analysts used to answer their questions included:

T1: Identify nodes/edges shared between graphs
T2: Identify nodes/edges present in one graph but not in others
T3: Identify when the direction of an edge differs between graphs
T4: Identify graph cycles
T5: Compare the confidence that algorithms place on edges
T6: Compare results changes from algorithm-input/parameter changes
T7: Compare causal graphs from ensembles of algorithms to those from individual algorithm results

In the causal discovery context, graph comparison contributes to feedback loops where the analysts change inputs to the causal discovery algorithms based on their interactions inside the graph comparison tools. Tasks that close the feedback loop and update causal discovery (abbreviated FT to distinguish from the internal tasks) include:

FT1: Specify edges that future iterations should ignore
FT2: Specify edges that are asserted to be present
FT3: Identify nodes to combine within a graph (common when dealing with related factors' variables, e.g., one-hot encoding)
FT4: Identify nodes that are noise and should be removed from consideration, additional variables to collect data for, or constructs to include (e.g., proxies of unobserved variables)
FT5: Construct ensembles from the results of several algorithms

These feedback loops are an important way that our graph comparison tool enables human-machine teaming; analysts' tacit knowledge and observations can be used as inputs to the discovery algorithms to refine and improve discovery on subsequent iterations.

The challenges of graph comparison are simpler in a causal discovery context than in a general-purpose context. There are 3 major differences. First, the range of things that must be compared is constrained by *reliable labels* in the causal discovery context. The labels come from the input data, and are generally shared between the different algorithm results. (Some algorithms may add or drop columns, but any node with a label from the input data represents the same thing.) This tightly constrains which graph elements must be compared to each other. Second, the types of comparisons done between nodes and edges are more constrained. Rather than comparing arbitrary attribute values, causal discovery attribute comparison is most often concerned with edge weight (sometimes absolute weight, and sometimes the difference between two weight). Third, the structural properties of most interest in the causal discovery context are more often *local* properties, while general graph comparison may look at global properties. Beyond the presence/absence of specific links, direction disagreements are the most important structural property for in the causal discovery context. Using causal discovery tasks to focus the graph comparison tasks also focused and simplified our interface.

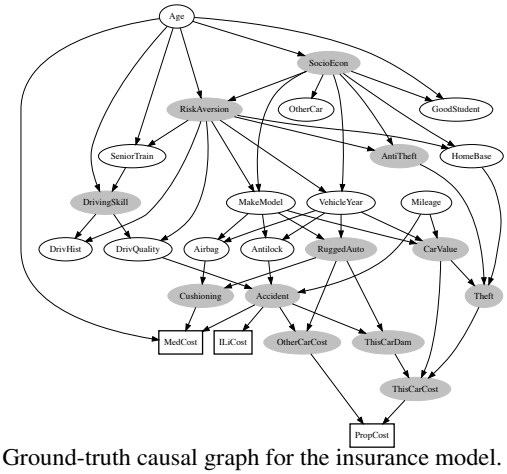


Fig. 2: Ground-truth causal graph for the insurance model. "Hidden" (grayed) would typically be unobservable for real-world insurance evaluations. Outcome nodes are rectangular.

4 Graph Comparison Visualizations

To illustrate the visualizations and how analysts can leverage them, we use the 'Insurance' causal model of car insurance risks [3][1]. The model and data samples come from BNLearn, an R package for bayesian network learning and inference [43]. The overall model tries to estimate the expected claim costs of an insurance holder. Although there are 27 nodes and 52 causal relationships (edges) in the source causal model (shown in Figure 2) , only 15 of the nodes are *observable* variables that would be accessible in real-world applications. The variables are a mixture of binary, categorical, ordinal and semi-continuous (i.e., integer numerical over a wide range). The variables are related to the insured vehicle (e.g., Mileage, VehicleYear, etc.), the insurance holder (e.g., Age, GoodStudent, etc.) and the potential outcomes of a claim (MedCost, PropCost, ILiCost). This dataset is used in all example visualizations in this paper.

We worked with a data analysis team and implemented our tool for graph comparison. It has two views that address the questions and supports the tasks identified earlier. These views were developed through many iterations with the analysis team, focusing on tasks and techniques the analysis team needed to achieve their goals. The first view compares several graphs at once in a *broad comparison* enabling an overview of many graphs (top in Figure 3; left in Figure 1). The second view compares two graphs directly in a *focused comparison* to get more detailed information about those graphs (bottom in Figure 3; right in Figure 1). The views share many interface elements to provide a consistent user experience. Figure 3 illustrates the two views and the controls/options made available in sidebars. (Figure 1 also shows the central component of each view.)

4.1 Shared Controls

Both views have a controls sidebar that support the tasks described in section 3. The controls are briefly described here (to highlight overlaps),

[1]The Insurance model was downloaded from `https://www.bnlearn.com/bnrepository/discrete-medium.html#insurance`

Fig. 3: Both views with respective toolbars, representing the insurance dataset. The top view is the broad comparison view, the bottom is the focused comparison view. The controls on the toolbars include (A) the button to navigate between views, graph similarity as (B) an algorithm map or (C) algorithm heatmap, (D) Venn diagram controls, (E) edge weight lters, (F) node lters by name or selection, (G) additional highlighting options for the broad comparison view, (H) node link lters in the focused view, (I) the ability to toggle node-label/tooltips visibility in the focused comparisons and (J) the ability to interact with Jupyter notebook state (these buttons are also present in the broad view, but clipped out for presentation purposes).

(a) Broad comparison with the Venn diagram set to intersection with the post-ensemble, highlighting edges that blacklisting removed. (d) Graph distance visualizations.

Fig. 4: Selected tool state comparing graphs before/after blacklisting

with view-speci c details provided in the description of the view itself.

The Venn comparison controls (Figure 3 control D) enable a direct comparison of graphs, using one algorithm's graph as a reference and comparing all other algorithms' graphs to it (tasks T1, T6, T7). This helps identify shared (or unique) nodes/edges. The Venn diagram control was the focused of many iterations. As the overall tool evolved and more causal questions were asked, the comparisons enabled by the Venn Diagram emerged as requirements. In early iterations it was a list of radio buttons, later a drop-down list. The Venn Diagram was eventually chosen because (1) it matched the language the analysts were eventually using and (2) it provided a compact representation of the possible con gurations. This type of iterative design tightly coupled with domain experts was used extensively in our development.

Some causal discovery algorithms include an edge weight to indicate relative con dence for the edge's presence. Edges can be ltered by their weights using the `Filter by edge weights' control (i.e., , control E; tasks T5, T6). This component is a histogram of weights with a selection band inlaid where the selection band can be manipulated to select a range of weights. To be most effective across many different graphs, the edge weights should be normalized before loading the graphs into the tool. However, because edge weight meaning varies by algorithm (especially when ensembling multiple results), there is no single normalization procedure. Edge normalization, therefore, is outside the scope of the tool itself (feedback task FT5).

Node lters may be set by regular expression in either view through the lter box (control F; tasks T1, T2). A list of current lters is accumulated below the lter box. Individual lters may be cleared by clicking on them or all lters can be cleared at once. Text-based entry was not originally an option (relying instead on direct selection), but it was added as the analysts often came with a focus in mind; providing text-based entry for ltering enabled them to avoid searching the graph for relevant nodes. This was later expanded to regular expressions as many graphs had nodes with related names (e.g., `cost' in the example graph is mentioned in several nodes).

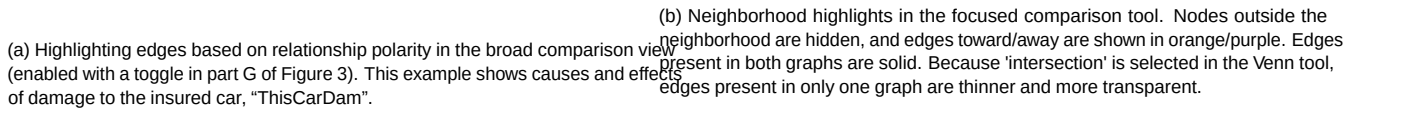
Navigating between the two views is done through the button (A), using the check-boxes above the graph columns in the broad view to select the two graphs to compare in the focused view. In the broad view, the button is disabled until two graphs are selected. Moving back to the broad view is always enabled in the focused view.

4.2 Broad Comparison View

The broad comparison view allows an analyst to compare many graphs at once, illustrated in Figure 3. This view shows all graphs in a matrix-like format where every graph is shown in its own column and every node has a row. By default, upstream edges are plotted to the right of nodes (downstream edges to the left), which helps analysts to distinguish cause and effect more easily (when control G is set to `compact view' all edges are placed on one side: more graphs may be seen at once but a single graph is more cluttered). Representing graphs this way enables analysts to compare edges related to a single node in many graphs (tasks T1, T2, T3) and is based on the TimeArcTrees layout [14]. The default Vertical node order is the spectral ordering [17, 32] of the union of all the graphs compared, but can be changed when the tool is instantiated. Spectral ordering sorts nodes using the second smallest Eigenvector of the Graph Laplacian (i.e., the Fielder Vector) [17]. Spectral ordering is related to many structural features of the graph [32] and tends to keep link distances short.

Showing all of the edges is often cluttered. The ltering options in Figure 3 parts E, F and G reduce the number of nodes and edges displayed (tasks T1, T4). The Venn diagram tool in part D changes the visibility/weighting of nodes and edges. Figure 1 shows a graph that has been ltered and the Venn Diagram control set to highlight edges shared with the rightmost graph. (The Venn diagram tool enables comparisons similar to those in Diffany [28]).

The broad comparison view also uses the node lter control. It also supports directly clicking on node names to highlight those nodes, and adding the current selection to the current lter (tasks T1, T2). When ltering by direct selection, the node and its neighbors are kept. The



(a) Highlighting edges based on relationship polarity in the broad comparison view (enabled with a toggle in part G of Figure 3). This example shows causes and effects of damage to the insured car, “ThisCarDam”.

(b) Neighborhood highlights in the focused comparison tool. Nodes outside the neighborhood are hidden, and edges toward/away are shown in orange/purple. Edges present in both graphs are solid. Because 'intersection' is selected in the Venn tool, edges present in only one graph are thinner and more transparent.

Fig. 5: Additional details available in the Broad and focused comparison views

neighbors are also kept to keep context of the lter for future analysis.

Much like ltering nodes or edges, analysts might need to lter the graphs they are comparing. To do this they can use the “Graph Plot supporting control (Figure 4b) to understand which graphs are similar (tasks T6, T7). The plot uses scikit-learn's multi-dimensional scaling (MDS) method with the previously mentioned graph distance metric to display all the graphs on a 2D plot [26, 37]. Points on this plot can be used to toggle the visibility of a graph. The “Graph Heatmap” control (Figure 4b lower) is an alternate way to explore graph similarity and lter out algorithms. It shows pairwise similarities between all graphs to summarize overall agreement (or disagreement). It is based on the same metric as graph order and darker colors indicate that more components are shared. A graph's lter state is toggled in the graph heatmap control by clicking on its name along either axis.

A collection of toggle controls also provides additional view con gurations (Figure 3, section G), speci cally informed by causal discovery. Much like the Venn diagram control, these options were developed during an iterative process with the analysts. This is not an exhaustive list of what was tried, but it represents con gurations that were consistently useful. The con guration controls are:

Sorting by metric: When unchecked, graphs are displayed in the order they are passed. When checked, order is based on similarity to the reference graph (task T6, T7). The default uses PageRank [34] to sort nodes and Kendall's Tau to compute similarity to the rst graph [24]. A custom function can also be provided.

Highlight edges using polarity: Edge polarity indicates if an input reenforces or inhibits a response. This option changes the edge style coding to emphasize disagreements (T3) and is similar to options used in other causal-reasoning tools [28, 46, 47]. (see Figure 5a for an example of this in use.)

Highlight globally shared edges: Finding which relationships algorithms agree on is a comparison. This option highlights the links that are shared between all graphs (including the direction) in blue (tasks T1, T7). This can be used to build a white-list or design lters that remove groups that are consistent between algorithms (see Figure 1, left side, for an illustration of this option).

Highlight shared links with different directions: Algorithms may agree that a relationship exists, but disagree on the relationship direction. This is expected since many algorithms re ne a Markov Equivalence class [18]. Edges where there are disagreements are a place experts can assert their knowledge. Highlighting disagreements makes them easy to nd (tasks T1, T3, T7). This option highlights the links that are shared between all graphs (regardless of direction) in purple (see Figure 1, left side) and is similar to region of interest techniques in tree comparisons tools [33, 40].

Compact View: When comparing many graphs, horizontal display space is quickly used up. The `compact view' option puts all edges on same side. This adds more graphs to the screen, but makes tracing edges more dif cult (task T1, T2).

A typical work ow using the broad comparison view includes:

1. Use node selection lters or the regular expression lter to narrow down the important nodes to examine.
2. Compare linkage structure to understand differences and similarities using lters, thresholds, and discovery-focused toggles.
3. Save tool state to a variable (e.g., for documentation).
4. Use the insights from interactions with the tool to update blacklists, re ne algorithms, identify where to collect more data, or try different graph parameters in causal discovery.

4.3 Focused Comparison View

The focused comparison tool enables detailed comparisons between two graphs, designed to enable an analyst to nd interesting sections of the graphs and explore structures in more detail. It is most similar in style to other causal exploration tools as it uses a traditional node-link diagram as its main representation [46, 47]. This tool uses 3 juxtaposed panels that each display a graph, as in Figure 3. The panels on the sides show input graphs. The panel in the middle displays the result of a set operation on the input graphs' nodes and edges (similar to Diffany [28]). The three panels use a shared spring-force embedding (similar to DyNet and Gasoline [13, 31]) and follow coordinated multiple views (CMV) conventions for zoom, pan, and picking.

All three panels react to the currently selected comparison state in the Venn diagram tool. The central panel directly represents the current Venn diagram selection. Elements on the side panels are emphasized if they are also present in the middle or de-emphasized if they are not (similar to the change highlighting in [1]). Changes to opacity, line style and stroke colors are used indicate overlap/exclusion between the central and side panels. These encodings provide an analyst with quick access to information about how a node interacts with other nodes in each of the causal graphs compared.

In the three main panels, the analysts can hover over a node to get more information about it and its neighbors (see Figure 5b). This context is captured by the node, its incident edges, and its immediate neighbors (tasks T1, T2, T3, T4 for short cycles); all other nodes and edges are lter out so the information is presented as cleanly as possible. In this focus view, edges are colored by their direction to distinguish causes from effects and highlight differences in direction between the graphs (if present). This lter is hover-based instead of selection based because it is intended to be used as a quick-check. The focused view also includes a node-degree lter that is not present in the broad comparison view (control H in Figure 3). This enables analysts to select high or low degree nodes to focus in on further, depending on their current line of investigation.

If the analyst does not have previous knowledge of the causal graph, they can use the focused comparison view to identify where to investigate. If they do have knowledge of the graph, the focused comparison view can be used to compare approximations to that causal graph. Using the focused comparison typically follows the following steps:

1. Select a pair of graphs. By convention, if a canonical graph exists, it is selected and placed on the left.

2. Specify a difference or intersection comparison for the center pane and look for an interesting structure.
3. Use lters and hover to understand the community structure (confounder, collider, etc.) and the differences between the graphs.
4. Save the tool state to a variable for future use.
5. Use the insights to update algorithm parameters or inputs.

5 Example Workflow

Using the insurance dataset and our graph comparison tool, one of the analysts did an extended analysis session. Their task was to identify algorithms/parameters that could recover the causal graph (which is known) so the ensemble could be applied where the graph is not known. This section describes an abstraction/simpli cation of their work ow over several sessions and how our tools impacted it. Their work ow is iterative, passing through both visualization views and the causal discovery algorithms multiple times and informs Figure 1.

The analyst starts by trying a variety of algorithms, selected based on their availability and a lack of known violations of preconditions within the data. Because their task is to evaluate the effectiveness of algorithms, they treat all of the variables as observable at rst and include all 27 in the analysis. After the algorithms have completed causal discovery, all of the graphs are loaded in the broad comparison view (see Figure 3). Observing the similarity between algorithms in the graph plot and graph heatmap in the upper right, it is clear that MMPC creates a graph that is signi cantly different from the other algorithms (tasks T1, T2). The broad comparison view shows that both MMPC and GES have more edges than PC or CCDR. Furthermore, MMPC has more symmetric edge pairs than CCDR (task T4), since there are more pairs of edges on both side of the nodes column, illustrating both an upstream and downstream relationship between a pair of nodes.

Comparing MMPC to an ensemble of all algorithms (task FT5) in the focused comparison tool (Figure 3), those symmetric pairs are more apparent (task T7). Further, selecting to show the `intersection' of the two graphs using the Venn diagram tool shows that MMPC does not contribute many *unique* edges to the ensemble (tasks T1, T2, T7). This is clear because the central region is very similar to the ensemble graph. This leads the analyst to decide to omit MMPC from future analysis.

Based on the meaning of the variables, the analyst observes that the ensemble (and other individual algorithm graphs) have edges with whose direction contradict the analysts intuition (task T3). For example, vehicle-related properties (car cost) causing driver properties (good student) or age being the effect of any other variables. (The examples in Figure 1 show graphs focused on driver properties.) Therefore, the analyst creates a series of blacklisted edges (task FT1) as an input to a new iteration of the causal discovery algorithm (except for MMPC, which is excluded from future iterations). After a few iterations, the blacklist speci es that there can be no edges from vehicle properties to people properties, no edges into age, and no edges out of the expected cost variables. (For algorithms that do not take blacklists and input, these edges are removed at output time.)

To examine how these restrictions impact the output, the graphs before and after blacklist (task T6) are visualized in the broad comparison view (Figure 4a). The after graphs are visibly simpler and have fewer symmetric edges (task T4), while the graph plot shows that each algorithm remains most similar to itself across iterations. This view shows that GES is the most affected by the blacklisting (Figure 4b). The two versions of GES (task T6) are moved to the focused comparison view and the analyst decides the changes are an improvement (tasks T2, T3).

Along with identifying blacklist edges (task FT1), an analyst may identify a set of causal relationships that must be included (task FT2) or identify additional features that logically may be in uencing indirect causal paths, that might appear as unlikely edges. For example, in the insurance example we know that there are causal paths from the observed variable (“DrivQuality”, the quality of the insured's driving) to an unobserved variable (“Accident”, severity of the accident) and from the unobserved variable (“Accident”) to another observed variable (“ILiCost”, inspection cost). If the analyst removes the unobservable variables from causal discovery, several algorithms identify this indirect causal path as a direct causal link from “DrivQuality” to “ILiCost” (see

(a) CCDr, PC, and GES without constructs.

(b) GES without and with constructs.

Fig. 6: Broad comparison view lter ed to focus on relationships that include the insured driver's ability (“DrivQuality”) and the inspection cost (“ILiCost”), with and without the “LikelihoodOfAccident” construct.

Ref?	Question	Broad Comparison	Focused Comparison
True	What is right?	Best precision/recall?	What is right/missed?
False	What is happening	Which graphs are similar?	What is similar?

Table 2: Questions the tools can answer with/without a reference graph.

Figure 6a) because the intermediary variable is missing (task T2).

Observing the algorithm outputs in either the broad or focused comparison views, analysts can identify potential constructs or latent variables that were not originally observed but could be represented using a proxy generated from observed variables (tasks FT3, FT4). For example, the analyst may see the common link from “DrivQuality” to “ILiCost” and, using their subject matter expertise, hypothesize that this may be an indirect path with a missing intermediary node such as the driver's likelihood of the type or severity of an accident and construct a proxy for this unobserved variable using a combination of driver characteristics (quality of driving, driving history, age, home-base location) and vehicle characteristics (whether it is older, has higher mileage, has anti-lock brakes) that may contribute to the likelihood or severity of an accident. In Figure 6b, we see that GES accurately identi es the causal path of “DrivQuality” ! “LikelihoodOfAccident” ! “ILiCost” instead of from “DrivQuality” to “ILiCost” directly.

As shown in Figure 7, the analyst can also use the focused comparison view to see improvement in causal relationships that do not directly incorporate the constructs that were added (tasks T5, T6). In this case the center panel illustrates the new causal relationships discovered by GES when the additional constructs were included. When the analyst compares the results of causal discovery from the GES algorithm with and without the constructs, there are several spurious links that are no longer included: i.e., a causal link from the vehicle's mileage to the inspection cost. The analysts continued through several more iteration cycles before settling on set of algorithms and parameters.

Fig. 7: Focused comparison view comparing GES before (left panel) and after (right panel) including constructs, filtered to causal relationships that include the driving quality of the insured driver ("DrivQuality") or the inspection cost outcome ("ILiCost"), examining the difference when the algorithm has access to the "LikelihoodOfAccident" construct.

6 Discussion

The graph-comparison tool can also be used to evaluate the effectiveness of causal discovery algorithms when there is a reference graph (i.e., a `ground-truth' graph). This ground truth is often known for synthetic datasets because it is part of the data curation process. It may also be created by a subject matter expert or be provided by a specific benchmark algorithm. In each cases, the fundamental task shifts from exploration of options to evaluation. Table 2 outlines the different questions each tool can address with/without a reference graph. The major difference when working with a reference graph is that comparisons are more directed; the focus is on what is missed (or correctly identified). In the broad comparison tool, the reference graph is passed as the first graph (or selected in the sidebar controls). When observing other graphs, the dotted lines then indicate errors instead of simply differences. In the focused comparison tool, the Venn diagram control can be used to directly ask for missing nodes/edges by selecting left-only or for extra edges by selecting the right-side only.

In the example work flow the focus was on comparisons between algorithms. However, ensemble refinement typically involves several feedback cycles in its own right. `Which algorithms should be used in the ensemble?' is the base question. Comparison between different algorithm combinations is similar to the discussion about blacklist/whitelist but looks at different combinations of algorithms instead the direct impact of parameters on specific algorithms. In our experience, ensemble evaluation is interleaved with an algorithm-by-algorithm exploration; individual algorithms may be removed from an ensemble based on its individual evaluation.

The tools were originally developed with examples from research papers with few nodes and relatively sparse edges. Applications often generated much larger graphs. Some applications had hundreds of nodes, resulting from using 1-hot (or -hot) encoding to represent variables for the discovery algorithms. This drove our team to develop workflows that included aggregating, selecting and thresholding causal discovery algorithm outputs prior to loading the graphs into our tools in addition to the selecting/thresholding done in the tools. In practice, 30-or-so node networks are practical for these tools.

7 Future Work

The graphs loaded into the broad comparison tool were essentially static. However, the broader workflow often involved ensembling different discovery algorithms and blacklisting edges. These techniques are entirely external to our tools, but would be beneficial additions. The merging technique shown in TimeArcTrees [14] cannot be applied directly (since it is sequence dependent and always removes the contributing graphs) but a similar technique may be if a sequence of causal graphs represents different causal behavior at different times in a time series. It may also be a means to build ensembles interactively.

We are looking at extending the focused comparison tool to more graphs. The focused comparison layout is a natural metaphor for comparison. However, the underlying analysis is not inherently limited to two graphs. The focused comparison tool provides more algorithmic support for comparison than the broad comparison tool (which relies more on the visual system to do comparisons), which makes extending it to more graphs appealing. Avenues of exploration include different layouts that would enable three or four graphs to be compared naturally and how to incorporate information-quality into the analysis and display (such as known ground truth or expert assertions).

As noted, the analyst workflow was simplified because the tool was integrated directly into in a Jupyter notebook. However, the integration with the notebook is current fairly superficial; variables can be passed in and a copy-paste mechanism allows state to flow back out. This forced a beneficial loose coupling, and related flexibility, with the analysis algorithm without directly interfacing with the analysis algorithms but being able to set Jupyter notebook variables directly would ease interactions. For example, the ability to export an edge-filter set to a python variable would enable more rapid iteration around black- or white-lists, without directly interfacing with the causal discovery algorithms (the interaction would still be mediated by the Jupyter environment and use cells to drive execution). This deeper integration would likely require changes to analysts' expectations and would present technical challenges for execution and reproducibility.

With a more complete integration, the overall process of causal graph creation could be more like taking notes on a dataset. The graph represents an operational hypothesis and interactions with the data are done in the context of that graph. This is akin to a "higher-order" view of the dataset [7]. Combined with algorithmic discovery, the techniques discussed in this paper lay a foundation for interactively constructing that view. A well integrated system, where discovery and analysis revolve around a shared causal graph would be a powerful interactive analysis tool. Combining our techniques with the simulation-results focused visualizations from Wang and Muelder [47] would be a strong step towards realizing Pearl's causal inference engine [35].

8 Conclusion

This paper has argued that graph comparison is essential for causal discovery. A human-in-the-loop is essential to building causal model because of the inherent limitations in causal discovery algorithms. Using tools that focused on graph comparison improved analysts workflows and helped create causal graphs that the analysts had more confidence in. We have found that tools can be built into an existing analysis environment (i.e., Jupyter notebooks) to make a more integrated workflow. This change in work flow helped analysts to move beyond descriptions of what had occurred to discovering why.

Acknowledgments

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

REFERENCES

[1] B. Bach, E. Pietriga, and J.-D. Fekete. Graphdiaries: Animated transitions andtemporal navigation for dynamic networks. *IEEE Transactions on Visualization and Computer Graphics*, 20(5):740–754, 2014.

[2] F. Beck, M. Burch, S. Diehl, and D. Weiskopf. A taxonomy and survey of dynamic graph visualization. *Comput. Graph. Forum*, 36(1):133–159, Jan. 2017. doi: 10.1111/cgf.12791

[3] J. Binder, D. Koller, S. Russell, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29(2):213–244, 1997. doi: 10.1023/A:1007421730016

[4] D. M. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3(Nov):507–554, 2002.

[5] D. Colombo and M. H. Maathuis. Order-independent constraint-based causal structure learning. *The Journal of Machine Learning Research*, 15(1):3741–3782, 2014.

[6] D. Dingen, M. van't Veer, P. Houthuizen, E. H. J. Mestrom, E. H. H. M. Korsten, A. R. A. Bouwman, and J. van Wijk. Regressionexplorer: Interactive exploration of logistic regression models with subgroup analysis. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):246–255, 2019.

[7] T. Dwyer. Network visualization as a higher-order visual analysis tool. *IEEE Comput. Graph. Appl.*, 36(6):78–85, Nov. 2016. doi: 10.1109/MCG.2016.117

[8] J. Ellson, E. Gansner, E. Koutsofios, S. North, and G. Woodhull. Graphviz and dynagraph – static and dynamic graph drawing tools. In M. Junger and P. Mutzel, eds., *Graph Drawing Software*, Mathematics and Visualization, pp. 127–148. Springer-Verlag, Berlin/Heidelberg, 2004.

[9] N. Elmqvist and P. Tsigas. Causality visualization using animated growing polygons. In *Proceedings of the IEEE Symposium on Information Visualization*, pp. 189–196. IEEE, New York City, NY, 2003.

[10] N. Elmqvist and P. Tsigas. Growing squares: Animated visualization of causal relations. In *Proceedings of the ACM Symposium on Software Visualization*, pp. 17–26. IEEE, New York City, NY, 2003.

[11] N. Elmqvist and P. Tsigas. Animated visualization of causal relations through growing 2d geometry. *Information Visualization*, 3(3):154–172, 2004.

[12] C. Glymour, K. Zhang, and P. Spirtes. Review of causal discovery methods based on graphical models. *Frontiers in Genetics*, 10:524, 2019. doi: 10.3389/fgene.2019.00524

[13] I. H. Goenawan, K. Bryan, and D. J. Lynn. DyNet: visualization and analysis of dynamic molecular interaction networks. *Bioinformatics*, 32(17):2713–2715, 05 2016. doi: 10.1093/bioinformatics/btw187

[14] M. Greilich, M. Burch, and S. Diehl. Visualizing the evolution of compound digraphs with timearctrees. *Comput. Graph. Forum*, 28(3):975–982, 2009. doi: 10.1111/j.1467-8659.2009.01451.x

[15] R. P. Grimaldi. *Discrete and Combinatorial Mathematics; An Applied Introduction*. Addison-Wesley Longman Publishing Co., Inc., USA, 4th ed., 1999.

[16] R. Guo, L. Cheng, J. Li, P. R. Hahn, and H. Liu. A survey of learning causality with data: Problems and methods, 2018.

[17] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using networkx. In G. Varoquaux, T. Vaught, and J. Millman, eds., *Proceedings of the 7th Python in Science Conference*, pp. 11 – 15. Pasadena, CA USA, 2008.

[18] A. Hauser and P. Bühlmann. Characterization and greedy learning of interventional Markov equivalence classes of directed acyclic graphs. *Journal of Machine Learning Research*, 13:2409–2464, 2012.

[19] J. M. Hofman, A. Sharma, and D. J. Watts. Prediction and explanation in social systems. *Science*, 355(6324):486–488, 2017. doi: 10.1126/science.aal3856

[20] A. Jaber, J. Zhang, and E. Bareinboim. Causal identification under markov equivalence, 2018.

[21] D. Kalainathan and O. Goudet. Causal discovery toolbox: Uncover causal relationships in python, 2019.

[22] M. Kalisch and P. Bühlmann. Causal structure learning and inference: A selective review. *Quality Technology & Quantitative Management*, 11(1):3–21, 2014. doi: 10.1080/16843703.2014.11673322

[23] M. Kalisch, M. Mächler, D. Colombo, M. H. Maathuis, and P. Bühlmann. Causal inference using graphical models with the R package pcalg. *Journal of Statistical Software*, 47(11):1–26, 2012.

[24] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

[25] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, eds., *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pp. 87 – 90. IOS Press, 2016.

[26] J. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

[27] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, July 1978. doi: 10.1145/359545.359563

[28] S. V. Landeghem, T. V. Parys, M. Dubois, D. Inzé, and Y. V. de Peer. Diffany: an ontology-driven framework to infer, visualise and analyse differential molecular networks. *BMC Bioinformatics*, 17(1):18, 2016. doi: 10.1186/s12859-015-0863-y

[29] M. Levandowsky and D. Winter. Distance between sets. *Nature*, 234(5323):34–35, 1971. doi: 10.1038/234034a0

[30] D. Margaritis and S. Thrun. Bayesian network induction via local neighborhoods. In *Advances in Neural Information Processing systems*, pp. 505–511, 2000.

[31] G. Micale, A. Continella, A. Ferro, R. Giugno, and A. Pulvirenti. Gasoline: a cytoscape app for multiple local alignment of ppi networks. *F1000Research*, 3:140–140, 2014. doi: 10.12688/f1000research.4537.2

[32] B. Mohar. The laplacian spectrum of graphs. In *Graph Theory, Combinatorics, and Applications*, pp. 871–898. Wiley, 1991.

[33] T. Munzner, F. Guimbretière, S. Tasiran, L. Zhang, and Y. Zhou. Treejuxtaposer: Scalable tree comparison using focus+context with guaranteed visibility. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, p. 453–462. Association for Computing Machinery, New York City, NY, USA, 2003. doi: 10.1145/1201775.882291

[34] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[35] J. Pearl. The seven tools of causal inference, with reflections on machine learning. *Commun. ACM*, 62(3):54–60, Feb. 2019. doi: 10.1145/3241036

[36] J. Pearl and D. Mackenzie. *The book of why: the new science of cause and effect*. Basic Books, New York City, NY, 2018.

[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[38] F. Pérez and B. E. Granger. IPython: a system for interactive scientific computing. *Computing in Science and Engineering*, 9(3):21–29, May 2007. doi: 10.1109/MCSE.2007.53

[39] T. Richardson. A discovery algorithm for directed cyclic graphs. In *Proceedings of the Twelfth International Conference on Uncertainty in Artificial Intelligence*, UAI'96, p. 454–461. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.

[40] O. Robinson, D. Dylus, and C. Dessimoz. Phylo.io : Interactive Viewing and Comparison of Large Phylogenetic Trees on the Web . *Molecular Biology and Evolution*, 33(8):2163–2166, 04 2016. doi: 10.1093/molbev/msw080

[41] S. Rufiange and M. J. McGuffin. Diffani: Visualizing dynamic graphs with a hybrid of difference maps and animation. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2556–2565, Dec. 2013. doi: 10.1109/TVCG.2013.149

[42] U. Schöning. Graph isomorphism is in the low hierarchy. *J. Comput. Syst. Sci.*, 37(3):312–323, Dec. 1988. doi: 10.1016/0022-0000(88)90010-4

[43] M. Scutari. Learning bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35(3):1–22, 2010. doi: 10.18637/jss.v035.i03

[44] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker. Cytoscape: A software environment for integrated models of biomolecular interaction networks. *Genome Research*, 13(11):2498–2504, Nov 2003.

[45] I. Tsamardinos, C. F. Aliferis, and A. Statnikov. Time and sample efficient discovery of markov blankets and direct causal relations. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 673–678. New York City, NY, 2003.

[46] J. Wang and K. Mueller. The visual causality analyst: An interactive interface for causal reasoning. *IEEE Transactions on Visualization and*

Computer Graphics, 22(1):230–239, 2016.

[47] J. Wang and K. Mueller. Visual causality analysis made practical. In *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 151–161. IEEE, New York City, NY, 2017.

[48] C. Zhang, B. Chen, and J. Pearl. A simultaneous discover-identify approach to causal inference in linear models. Technical Report R-491-L, url = http://ftp.cs.ucla.edu/pub/stat_ser/r491-L.pdf, Department of Computer Science, University of California, Los Angeles, CA, 2019. Extended version of paper accepted to the Proceedings of the Thirty-fourth AAAI Conference on Artificial Intelligence (AAAI-2020).

[49] J. Zhang. On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. *Artif. Intell.*, 172(16–17):1873–1896, Nov. 2008. doi: 10.1016/j.artint.2008.08.001

[50] X. Zheng, B. Aragam, P. Ravikumar, and E. P. Xing. Dags with no tears: Continuous optimization for structure learning, 2018.