Press ECCS to Doubt (Your Causal Graph)

Markos Markakis
MIT CSAIL
Cambridge, MA, USA
markakis@mit.edu

Ziyu Zhang
MIT CSAIL
Cambridge, MA, USA
sylziyuz@mit.edu

Rana Shahout
Harvard University
Cambridge, MA, USA
rana@seas.harvard.edu

Trinity Gao
MIT CSAIL
Cambridge, MA, USA
trinityg@mit.edu

Chunwei Liu
MIT CSAIL
Cambridge, MA, USA
chunwei@mit.edu

Ibrahim Sabek
University of Southern California
Los Angeles, CA, USA
sabek@usc.edu

Michael Cafarella
MIT CSAIL
Cambridge, MA, USA
michjc@csail.mit.edu

ABSTRACT

Techniques from the theory of causality have seen extensive use in natural and social sciences, since they allow scientists to explicitly model assumptions and draw quantitative causal conclusions. More recently, causality has also gathered interest in many computer science sub-fields, including machine learning and systems. A causal model is usually represented as a causal graph, often automatically discovered from available data. For problems for which running a full constraint-based causal discovery algorithm and correctly orienting all edges is computationally intractable, automatically generated causal graphs are prone to error, calling for expensive manual graph verification. Understanding which parts of a causal graph have the largest impact on downstream results is essential for expediting this graph verification process.

In this work, we present ECCS – a framework for **E**xposing **C**ritical **C**ausal **S**tructures within a causal graph, with respect to a given Average Treatment Effect (ATE) calculation. We formalize the Interactive Causal Graph Verification problem, in which user *judgments* about edges in the causal graph are solicited sequentially, with the goal of minimizing the absolute error in the ATE of interest (without advance access to its ground-truth value). We present three algorithms to solve this problem. Based on a preliminary evaluation, our best-performing algorithm, AdjSetEdit, can solicit a sequence of 10 user judgments that outperforms a randomized such sequence by more than 60%, with time complexity linear in the number of data points and polynomial in the number of variables.

CCS CONCEPTS

• **Computing methodologies → Causal reasoning and diagnostics**.

KEYWORDS

Causality, Causal Graph, Causal Graph Verification

GUIDE-AI '24, June 14, 2024, Santiago, AA, Chile

ACM ISBN 979-8-4007-0694-3/24/06
https://doi.org/10.1145/3665601.3669842

ACM Reference Format:
Markos Markakis, Ziyu Zhang, Rana Shahout, Trinity Gao, Chunwei Liu, Ibrahim Sabek, and Michael Cafarella. 2024. Press ECCS to Doubt (Your Causal Graph). In *Governance, Understanding and Integration of Data for Effective and Responsible AI (GUIDE-AI '24), June 14, 2024, Santiago, AA, Chile.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3665601.3669842

1 INTRODUCTION

Causal reasoning has greatly aided scientists in posing, discussing and testing hypotheses about cause-effect relationships in different fields including medicine [57], economics [24], biology [9] and social sciences [45]. In recent years, causality has also piqued the interest of computer scientists. For example, machine learning researchers leverage it to provide additional structure and trustworthiness to machine learning problems (leading to research areas like causal representation learning [40] and causal reinforcement learning [13, 61]), while computer systems researchers use it to better understand and debug large complicated systems [6, 12, 31].

In particular, Pearl's framework for causality [33] is canonical for computing the *Average Treatment Effect (ATE)* of a treatment variable T on an outcome variable Y (see more background on causal reasoning in Section 2). This framework leverages a *causal model* for the problem at hand, represented as a *causal graph* with a node for each problem variable. Sometimes, constructing a reference causal graph can itself be the end goal [9, 57]. However, our work focuses on another important setting, where the primary goal is to draw robust and certain conclusions based on existing observations [12, 39]. In other words, the causal graph is leveraged to ensure that downstream ATEs are calculated correctly from observational data by indicating a sufficient *adjustment set*: a set of variables to adjust for to avoid structural biases such as *confounding bias* - where variables sharing a common cause erroneously appear to cause each other [33]. The accuracy of the ATE is, therefore, a direct result of the quality of the causal graph.

However, high-quality causal graphs are difficult to obtain at scale. Graphs are most often either *hand-crafted* by experts or automatically *inferred* from available data – a process called *causal discovery*. Even though causal discovery algorithms are more scalable than manual construction, the graphs they generate are not perfect. Depending on the choice of algorithm, the generated graph may be *incomplete*, if the available data is not sufficient to determine the direction of all causal relationships [47, 48]. Even worse, depending on the requirements of the causal discovery algorithm

(e.g. independent noise terms [43]) and possible biases in the available data (e.g. selection bias), the generated graph may be *incorrect* with respect to the actual data generation process [26]. Even small mistakes in a causal graph can have a large impact on downstream results, if they imply a different adjustment set.

Thus, automatically-derived graphs are still *comprehensively verified* before use [10]. Manual, expert-driven verification is natural, but it is only practical when graphs are *small*, up to to a few dozen variables [21, 37]. Even then, it is time-consuming and tedious. However, many interesting domains (e.g. large distributed systems, social networks, and neural networks) involve many more causally related variables, and possibly a repeated need to create and verify graphs over them. For example, drawing causal conclusions from distributed system logs is essential for efficient fault localization and response, but it can require a causal model over hundreds of log-derived variables [31]. Can we make expert-driven causal graph verification viable in these domains?

In this paper, we present ECCS – a human-in-the-loop framework for **E**xposing **C**ritical **C**ausal **S**tructures in a given causal graph, from the perspective of a particular ATE that the user is interested in calculating. The key insight is that, while overall graph fidelity is important, **not all parts of a causal graph affect a particular ATE calculation**, as previous work has also studied from a different perspective [15]. This is true because not every causal graph inaccuracy will alter the adjustment set, and it is the adjustment set that determines the ATE. The expert verifier's time is best spent prioritizing the most impactful parts of the graph.

The goal of ECCS is therefore to **solicit user feedback about parts of the causal graph in order of decreasing importance** and to incorporate such feedback into the graph, efficiently increasing user confidence in the correctness of the ATE of interest. Over a series of *interaction rounds*, the user can iteratively receive suggestions from ECCS and refine their causal graph, thus increasing the reliability of the conclusions drawn from it. We present three preliminary algorithms for generating such suggestions in ECCS.

In summary, we make the following contributions:

- We present the *Interactive Causal Graph Verification* problem, which seeks to efficiently use an expert verifier's time to refine a causal graph, given an ATE query of interest.
- We introduce three strategies for this problem: (1) SingleEdit, which compares all single-edge graph edits; (2) HeuristicEdit, which uses the A* algorithm to explore possible causal graphs; and (3) AdjSetEdit, which converts impactful adjustment set edits to graph edits.
- We evaluate our strategies and find that SingleEdit and AdjSetEdit outperform a randomized baseline by over 60% on the absolute relative error in the ATE of interest after 10 user judgments. Invoking AdjSetEdit also offers an interactive experience, requiring 1.12 seconds on average.

In the remainder of this paper, we will first provide some background on causality (Section 2), before presenting the Interactive Causal Graph Verification problem and the user interaction model of ECCS (Section 3). We then present our three strategies (Sections 4-6) and evaluate them (Section 7). We discuss some related work (Section 8) before concluding with future directions (Section 9).

2 BACKGROUND

In this work, we adopt the conventional framework for causal inference developed by Pearl [33]. We next introduce our notation and summarize some fundamental causal inference concepts.

Notation. Uppercase letters denote variables (e.g. V) and lowercase letters their values (e.g. $V = v$). $V \rightarrow W$ stands for a directed edge from V to W. Boldface denotes sets of variables, edges or edge edits. Calligraphic font denotes causal graphs (e.g. $\mathcal{G}$). For directed graphs, we distinguish a *path* from a *directed path*.

The ATE. The causal effect of a treatment T on an outcome Y may be distorted by *confounding variables* (or *confounders*) $\mathbf{Z}$ which influence both T and Y [33, 34]. For example, age may affect both a patient's likelihood of surgery and their odds of recovery. To sidestep confounders, one can collect *interventional* data, where T is not influenced by $\mathbf{Z}$ because it is explicitly assigned - e.g. through a randomized controlled trial (RCT). However, intervention on the entire system may not always be ethical, tractable, or feasible. Thus, we focus on *observational* average treatment-effects (ATEs).

Definition 1 (ATE from Observational Data [33]). *Given a treatment T, an outcome Y, and a valid adjustment set $\mathbf{Z}$,*

$$ATE(T, Y) = \mathbb{E}_Z \left[\mathbb{E}\left[Y | T = 1, \mathbf{Z} = z\right] - \mathbb{E}\left[Y | T = 0, \mathbf{Z} = z\right] \right]$$

To accurately calculate the ATE, $\mathbf{Z}$ needs to correctly adjust for confounding biases and not introduce spurious causal relationships. We discuss more relevant technical details in Section 6.1.

Causal Model A *casual model* expresses each variable V_i as a function of its direct causes (its *parents*) $\mathbf{P}_i$. These functions can be arbitrary, but they are often assumed to be linear [33, 48]. To obtain a causal model, we can either rely on expert input, or use data-driven *causal discovery* algorithms [14, 54], which generally rely on conditional independence calculations and/or information-theoretic or statistical scoring of causal models.

Causal Graph A causal model can be represented as a *causal graph* [32]: a directed acyclic graph (DAG) with one node per variable, where $V_i \rightarrow V_j$ implies $V_i \in \mathbf{P}_j$. In this context, we may use "variable" to refer to the corresponding node in the causal graph. One way to identify a valid adjustment set is to apply the *backdoor criterion* [33] to a causal graph (more details in Section 6.1).

3 OVERVIEW OF ECCS

3.1 Problem Definition

We start with a given observational dataset D with variables $\mathbf{V}$, and a specific ATE query of interest to the user: $ATE(T, Y)$, with $T, Y \in \mathbf{V}$. We assume a combination of user and domain such that the user is always capable of causal *judgments*: that is, given a pair of variables from $\mathbf{V}$, the user can specify whether a direct causal relationship exists and in which direction (but not its strength).

As we saw in Section 2, computing $ATE(T, Y)$ from an observational dataset like D requires a valid adjustment set, which can be derived from a causal graph using the backdoor criterion. Although the user *could* fully specify the causal graph $\mathcal{G}_{real}$, this would require $O(|\mathbf{V}|^2)$ judgments, one for every possible pair of problem variables. This complexity is impractical for large $\mathbf{V}$.

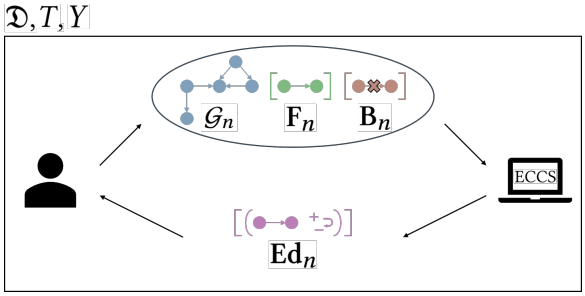


Figure 1: The user interaction model of ECCS.

Instead, we assume that the user can obtain an automatically-generated causal graph $\mathcal{G}_1$ of unknown correctness over the variables in $\mathbf{V}$. The expert could verify $\mathcal{G}_1$ fully, but this clearly requires the same number of judgments as constructing the graph. Moreover, many of the judgments made by the user during such an exhaustive verification will not impact $ATE(T, Y)$, because they will not be relevant when assessing the backdoor criterion.

Our goal is therefore to solicit feedback from the expert verifier efficiently, by only requiring them to produce judgments that would actually affect $ATE(T, Y)$. We do this over a series of ECCS-user *interaction rounds*, with ECCS soliciting one or more judgments per round. To keep each round interactive, ECCS must be fast in determining which judgment(s) to solicit in each round. Formally, the problem that ECCS aims to solve is the following:

Problem 1 (Interactive Causal Graph Verification). *Let J be a budget of total judgments, to be made over possibly several ECCS-user interaction rounds. Ensuring that ECCS's latency does not exceed L during each round, solicit judgments so as to produce a graph $\mathcal{G}_{final}$ that minimizes the absolute ATE Error:*

$$\mathcal{G}_{final} = \arg\min_{\mathcal{G}} |ATE_{\mathcal{G}}(T, Y) - ATE_{\mathcal{G}_{real}}(T, Y)| \quad (1)$$

Note that $\mathcal{G}_{real}$ and therefore $ATE_{\mathcal{G}_{real}}(T, Y)$ are not accessible during the interactive process, so it is not possible to directly minimize Objective 1. Another algorithmic barrier is the super-exponential number of possible DAGs on $|\mathbf{V}|$ variables [36, 44]. Therefore, ECCS must use different strategies to *explore* and *score* sets of edits. In Sections 4-6, we will present three such strategies.

3.2 Interaction Model

Concretely, the interaction model of ECCS is presented in Figure 1. Within a context of D, T and Y, users interact with ECCS iteratively. In a given interaction round n, the user provides ECCS with a causal graph $\mathcal{G}_n$ and two lists of directed edges: a list of *FIXED* edges $\mathbf{F}_n$, which should not be removed from the graph, and a list of *BANNED* edges $\mathbf{B}_n$, which should not be added to the graph. These lists let the user specify parts of the graph they are confident about, so that ECCS avoids spending computational resources evaluating the impact of editing these parts[1]. Edges in $\mathcal{G}_n$ but not $\mathbf{F}_n$ are called *PRESENT*, while edges in neither $\mathcal{G}_n$ or $\mathbf{B}_n$ are called *ABSENT*.

[1]Although ECCS receives these lists as inputs, the front-end user experience need not require manual curation of such lists - for example, we can have the user double-click an edge to mark it as *FIXED* in a graphical representation of the causal graph.

Algorithm 1 Suggest a single-edge edit that produces a maximally different $ATE(T, Y)$.

```
function SingleEdit(G, F, B)
    ate ← ATE(G)
    edits ← OneStepEdits(G, F, B)
    return arg max_{e∈edits} |EditAndGetATE(G, F, B, [e]) − ate|
```

Algorithm 2 Compute one-step edge edits from $\mathcal{G}$ based on specified fixed edges $\mathbf{F}$ and banned edges $\mathbf{B}$. $\mathbf{V}$ is the set of nodes of $\mathcal{G}$.

```
function OneStepEdits(G, F, B)
    edits = []
    for (V_i, V_j) in V × V
        if IsPresent(G, F, B, V_i, V_j)
            edits += (V_i, V_j, Remove)
        else if IsAbsent(G, F, B, V_i, V_j)
            if IsAbsent(G, F, B, V_j, V_i)
                edits += (V_i, V_j, Add)
            else
                edits += (V_j, V_i, Flip)
    return edits
```

For simplicity of exposition, we assume that for each edge in $\mathbf{F}_n$, the reverse edge is in $\mathbf{B}_n$. We also assume that for each edge in $\mathbf{B}_n$, its reverse is in either $\mathbf{F}_n$ or $\mathbf{B}_n$. These two conditions jointly reflect the fact that users have either already produced a judgment for the causal relationship between a pair of variables or not.

ECCS then generates a set of suggested causal graph edits $\mathbf{Ed}_n$ respecting $\mathbf{F}_n$ and $\mathbf{B}_n$. Each suggested edit consists of an directed edge and an operation: Add, Remove, or Flip. The edits in $\mathbf{Ed}_n$ are presented to the user sequentially. For each suggested edit, the user can revise the graph, fixed list and banned list according to how the suggestion compares to their known $\mathcal{G}_{real}$. Depending on the chosen strategy and the user's reaction to each suggested edit, ECCS may start a new interaction round $n + 1$ after each judgment, dropping any non-inspected elements of $\mathbf{Ed}_n$. Round $n + 1$ is also started once $\mathbf{Ed}_n$ has been fully inspected.

3.3 Auxiliary Functions

We now introduce some auxiliary functions. We assume that the dataset D, the treatment T and the outcome Y are fixed as context.

(1) IsFixed($\mathcal{G}$, $\mathbf{F}$, $\mathbf{B}$, V_i, V_j) returns `True` if and only if $V_i \to V_j$ is in $\mathcal{G}$ and in $\mathbf{F}$. We also equivalently define IsPresent, IsAbsent and IsBanned.
(2) ATE($\mathcal{G}$) uses $\mathcal{G}$ to compute the ATE of T on Y in D.
(3) IsAcceptable($\mathcal{G}$, $\mathbf{F}$, $\mathbf{B}$): Returns `True` if and only if $\mathcal{G}$ is a DAG that includes T, Y, all edges in $\mathbf{F}$ and no edges in $\mathbf{B}$. For a fixed tuple of ($\mathbf{F}$, $\mathbf{B}$), graphs $\mathcal{G}$ for which this function returns `True` will be called *acceptable* graphs.
(4) EditAndGetATE($\mathcal{G}$, $\mathbf{F}$, $\mathbf{B}$, $\mathbf{e}$) Given a set $\mathbf{e}$ of causal graph edits, applies the edits to a copy of $\mathcal{G}$ to obtain $\mathcal{G}'$ and then calls IsAcceptable($\mathcal{G}'$, $\mathbf{F}$, $\mathbf{B}$). If this call returns `False`, then EditAndGetATE returns `None`. Else, it returns ATE($\mathcal{G}'$).

4 STRATEGY 1: SINGLE EDGE EDIT

The first strategy, called SingleEdit, takes a "sensitivity" approach to the problem at hand: it focuses on finding the single-edge edit which, when applied to the input graph $\mathcal{G}$, will result in the *maximum absolute change* in $ATE(T, Y)$.

We believe that this greedy-like approach is valuable for our problem, assuming that the initial graph $\mathcal{G}_1$ gives a reasonable hint about $\mathcal{G}_{real}$. If the suggested edit is in line with $\mathcal{G}_{real}$ and the user accepts it, then this strategy helped take the most impactful correct step away from the initial graph. If the suggested edit is instead incorrect, then this strategy effectively "tightened" the ATE region under consideration by eliminating the point furthest away among the possible graphs close to the initial graph. When combined over the course of several interaction rounds, taking maximally large steps towards the right answer and reducing the allowed size of future steps helps converge to the correct value of $ATE(T, Y)$.

This strategy is listed in Algorithm 1. In a given iteration n, it starts with the current graph $\mathcal{G}_n$ and user specifications $\mathbf{F}_n, \mathbf{B}_n$. The algorithm computes OneStepEdits($\mathcal{G}_n, \mathbf{F}_n, \mathbf{B}_n$), which we will explain next. For each edit e returned by OneStepEdits, Algorithm 1 tries the edit and obtains a tentative ATE value ate = EditAndGetATE($\mathcal{G}_n$, $\mathbf{F}$, $\mathbf{B}_n$, $[e]$). Finally, it returns the edit that yields the maximum absolute ATE difference to solicit user judgment.

OneStepEdits (Algorithm 2) constructs a set of edits by checking the following conditions for each node pair in $\mathcal{G}$, $(V_i, V_j) \in \mathbf{V} \times \mathbf{V}$:

(1) If $V_i \to V_j$ is *PRESENT*, (V_i, V_j, Remove) is added.
(2) If $V_i \to V_j$ is *ABSENT*, (V_i, V_j, Add) is added if the edge $V_j \to V_i$ is also *ABSENT*, otherwise (V_j, V_i, Flip) is added.

From a time complexity standpoint, each SingleEdit invocation calls EditAndGetATE once for each of the $O(|\mathbf{V}|^2)$ outputs of OneStepEdits. Within EditAndGetATE, the call to ATE performs OLS linear regression, with complexity $O(|\mathsf{D}||\mathbf{V}|^2+|\mathbf{V}|^3)$. The overall complexity of SingleEdit is therefore $O(|\mathsf{D}||\mathbf{V}|^4 + |\mathbf{V}|^5)$.

5 STRATEGY 2: EXPLORING EDGE EDITS WITH THE A-STAR ALGORITHM

The fact that SingleEdit only considers single-edge edits means it can get stuck in local minima. For example, there are cases where adding *two edges* would have affected the ATE of interest by creating a new path in the causal graph, but no single edge edit can affect the ATE. Since SingleEdit maximizes the ATE impact of *an individual edge edit* at a time, it will overlook such cases.

Our next strategy, HeuristicEdit (Algorithm 3), changes the way suggested edits are scored, by relying on a heuristic search to identify edges that may be "lower-impact" individually, but which lie along a promising "edit path". Heuristic searches have traditionally been a critical tool to tackle computationally hard problems - for example, simulated annealing for the traveling salesperson problem [1] or evolutionary algorithms for feature selection [2].

At a high level, Algorithm 3 leverages the A* algorithm [16] over the space of possible causal graphs $\mathbf{\Gamma}$, starting from an input graph $\mathcal{G}$, and suggests the edge edit(s) most common among the "best-performing" causal graphs.

Each node in $\mathbf{\Gamma}$ is a possible causal graph over the variables in $\mathbf{V}$ that respects the fixed list $\mathbf{F}$ and the banned list $\mathbf{B}$. Each edge in $\mathbf{\Gamma}$ is bidirectional ($\mathcal{G}_m \leftrightarrows \mathcal{G}_n$) and implies that there is a pair of variables $(V_i, V_j) \in \mathbf{V} \times \mathbf{V}$ such that $\mathcal{G}_m$ and $\mathcal{G}_n$ only differ by $V_i \to V_j$. Each directed edge in the search graph $\mathbf{\Gamma}$ has a *type*, consisting of a causal graph edge and an operation. For example, assume that $V_i \to V_j$ is absent from $\mathcal{G}_m$ and present in $\mathcal{G}_n$. Then the search graph edge $\mathcal{G}_m \to \mathcal{G}_n$ has type $(V_i, V_j, addition)$, while the search graph edge $\mathcal{G}_m \leftarrow \mathcal{G}_n$ has type $(V_i, V_j, deletion)$. Note that the neighbors of $\mathcal{G}$ in $\mathbf{\Gamma}$ correspond to the outputs of OneStepEdits($\mathcal{G}$,$\mathbf{F}$, $\mathbf{B}$) if we ignore edits with type Flip. Our algorithm starts with $\mathcal{G}$ and explores towards the best point in $\mathbf{\Gamma}$. We define "best" below.

Algorithm 3 Suggest single-edge edits using heuristic search.

```
function HeuristicEdit(G, F, B, budget, m, k)
    edits = []
    frontier = min PriorityQueue
    frontier.push(G, 0)
    predecessor = {}
    while frontier is Not empty And |visited| < budget
        G' ← frontier.pop()
        if G' had been visited
            continue
        edits ← Filter(OneStepEdits(G', F, B), Not Flip)
        neighbors ← Filter(G'.Apply(edits), Not visited)
        for G'' ∈ neighbors
            neighbor_score = Φ(G) − Φ(G'')
            predecessor[G''] = G'
            frontier.push(G'', neighbor_score + (h(G'') ≡ 2Φ(G'')))
        MarkVisited(G')
    for G' ∈ k highest Φ graphs encountered
        Record common_edge_edits by counting the type of edge edits encountered on the path from G to G' using predecessor.
    return m edges from common_edge_edits with largest counts.
```

We assign each causal graph $\mathcal{G}$ a potential function.

$$\Phi(\mathcal{G}) = |ATE_{\mathcal{G}} - ATE_{init}| + \gamma \cdot quality(\mathcal{G}). \tag{2}$$

The *quality* uses heuristic measures of the candidate causal graph, such as sparsity, to regularize the search. In our current implementation, we use the closeness of the sparsity to a tree. γ decides the strength of the regularization. In future work, the potential function can be further elaborated to take into account the quality of the model's fit to the data or other similar confidence measures. Let $\mathcal{G}^\star = \arg\max_{\mathcal{H} \in \mathbf{\Gamma}} \Phi(\mathcal{H})$, but note that $\mathcal{G}^\star$ is not tractably known.

In $\mathbf{\Gamma}$, the weight of an edge $w(\mathcal{G}_m \to \mathcal{G}_n)$ is defined as $\Phi(\mathcal{G}_m) - \Phi(\mathcal{G}_n)$. This way, minimizing the cost of a path corresponds to maximizing the potential of the frontier node of the path (by telescoping). During a search starting from $\mathcal{G}$, when the frontier of the search path is $\mathcal{G}'$, A^* requires an estimate $h(\mathcal{G}'')$ of the cost of the *remaining path* between $\mathcal{G}'' \to \mathcal{G}^\star$ for each neighbor $\mathcal{G}''$ of $\mathcal{G}'$, in order to decide which neighbor to explore next. Let $h(\mathcal{G}'') = 2(\Phi(\mathcal{G}'') - \max_{\mathcal{G} \in \mathbf{\Gamma}} \Phi(\mathcal{G}))$. Since the remaining path cost is $\Phi(\mathcal{G}'') - \Phi(\mathcal{G}^\star)$, $h(\cdot)$ does not overestimate its value. Therefore, the heuristic strategy is *admissible* [16].

We set a computational budget *budget* and run A* over $\mathbf{\Gamma}$ for *budget* steps. That is, expand *budget* neighbors. At the end, the algorithm takes the k graphs with the greatest potentials explored, and examines the paths from the initial graph to these k candidates. We count the number of each type of search graph edge encountered and return the m most common types as the suggested causal graph edits to the user, one at a time. Only after the user has issued the corresponding m judgments is a new interaction round started, and HeuristicEdit is invoked again at that point to replenish the list of suggested causal graph edits. The time complexity of each invocation is $O(budget \log(budget) \cdot (|\mathsf{D}||\mathbf{V}|^2 + |\mathbf{V}|^3))$.

6 STRATEGY 3: ADJUSTMENT SET EDIT

The HeuristicEdit strategy showcased one way of ensuring that SingleEdit does not get stuck in local minima: explore "farther

away" from the starting graph in order to score the impact of each individual edge edit. However, HeuristicEdit still fundamentally "draws" suggested edits from the full space of possible single-edge graph edits, which is quadratic in the number of variables $|\mathbf{V}|$. Our final strategy, AdjSetEdit, instead works over the space of possible single-variable *adjustment set* edits, which is linear in $|\mathbf{V}|$.

The intuition here is to take a similar "sensitivity" approach as SingleEdit but apply it to the adjustment set instead of the graph edges. Namely, this strategy tries to identify the single most impactful *adjustment set edit* instead of the single most impactful edge edit, in terms of producing a large absolute ATE difference.

The top-level AdjSetEdit algorithm is presented in Algorithm 4. It takes the same inputs as Algorithm 1 and starts similarly, by calculating the ATE on $\mathcal{G}$. It then finds a valid adjustment set for $\text{ATE}_{\mathcal{G}}(T, Y)$ using existing algorithms [49]. The algorithm then considers each variable that *could* be in the adjustment set ($\mathbf{V} \setminus \{T, Y\}$) and finds edge edits that toggle its current adjustment set membership using MapRemoval (Section 6.2) or MapAddition (Section 6.3). Finally, we return the set of edits that yields the maximum absolute ATE difference among all sets considered above.

The algorithmic challenge is mapping an adjustment set edit to a set of causal graph edits to suggest to the user. We explain *one* approach in MapAddition and MapRemoval for the addition and deletion of one variable respectively. In general, multiple different sets of causal graph edits can achieve the same inclusion or exclusion of a specific variable to/from the adjustment set. We leave the full characterization of this problem to future work.

Note that, like HeuristicEdit, this strategy can suggest several (but still a small number of) edge edits per interaction round, which will be presented to the user one at a time. However, unlike the edits produced by HeuristicEdit, the set of edits produced by AdjSetEdit only makes sense *as a collection*, since all edits are required to achieve the desired effect on the adjustment set. Therefore, we continue presenting the suggested edits to the user sequentially only as long as the user's judgments agree with the suggestions. If the user disagrees with a suggested edit (e.g. we suggest removing $V_i \to V_j$, but the user determines that this edge should remain part of the causal graph and fixes it), we discard any remaining suggested edits and start a new interaction round by invoking AdjSetEdit again.

In terms of time complexity, AdjSetEdit identifies a set of edits for each of $O(|\mathbf{V}|)$ variables by calling MapRemoval or MapAddition. As we will see in Sections 6.2- 6.3, these algorithms have complexity $O(|\mathbf{E}| + |\mathbf{V}|)$ and $O(|\mathbf{E}|)$ respectively, where $\mathbf{E}$ is the set of edges of $\mathcal{G}$, meaning that generating the sets of edits has complexity $O(|\mathbf{V}| \times (|\mathbf{E}| + |\mathbf{V}|)) = O(|\mathbf{V}|^3)$. For each of the $O(|\mathbf{V}|)$ sets of edits, AdjSetEdit calls EditAndGetATE, with complexity $O(|\mathbf{D}||\mathbf{V}|^2 + |\mathbf{V}|^3)$. The overall complexity of AdjSetEdit is therefore $O(|\mathbf{D}||\mathbf{V}|^3 + |\mathbf{V}|^4)$, a factor of $|\mathbf{V}|$ better than SingleEdit.

6.1 Additional Causality Preliminaries

In order to present MapRemoval and MapAddition, we introduce some additional background on the theory of causal graphs. We begin by extending the notion of a parent:

Definition 2 (Ancestors and Descendants). *From each* ancestor *of V_i ($W \in \mathbf{An}(V_i)$) there is a directed path* to *V_i. Similarly, to each* descendant *of V_i ($W \in \mathbf{De}(V_i)$) there is a directed path* from *V_i. We let $V_i \not\in \mathbf{An}(V_i)$ and $V_i \not\in \mathbf{De}(V_i)$.*

Algorithm 4 Suggest a set of edits that produces a maximally different $ATE(T, Y)$, among sets of edits generated by considering adjustment set membership.

```
function AdjSetEdit(G, F, B)
    ate ← ATE(G)
    edit_sets ← []
    base_adj_set ← AdjSet(G)
    for V in G.nodes \ {T, Y}
        if V ∈ base_adj_set
            edit_sets += MapRemoval(G, F, B, V)
        else
            edit_sets += MapAddition(G, F, B, V)
    return arg max_{e ∈ edit_sets} |EditAndGetATE(G, F, B, e) − ate|
```

Algorithm 5 Suggest causal graph edits that would remove a variable from the adjustment set.

```
function MapRemoval(G, F, B, V)
    edits ← []
    if V ∈ An(T)
        P ← DirPaths(V, T)
        for p ∈ P
            (V_i → V_j) ← the first Not Fixed edge in p.
            if (V_i → V_j) does not exist
                return []
            edits += (V_i, V_j, Remove)
    T ← De(T) ∪ {T}
    Sort T by DirPathLengthFrom(T) in ascending order.
    for W ∈ T
        if IsAbsent(G, F, B, W, V)
            if IsAbsent(G, F, B, V, W)
                edits += (W, V, Add)
            else
                edits += (V, W, Flip)
                return Deduplicate(edits)
    return []
```

Given these notions, we can next present the idea of path blocking and the full backdoor criterion, as mentioned in Section 2:

Definition 3 (Path Blocking [33]). ***Path** p **is blocked** by nodes $\mathbf{B}$ if and only if:*

(1) p has a chain $i \to m \to j$ or fork $i \leftarrow m \to j$ s.t. $m \in \mathbf{B}$; or
(2) p has a collider $i \to m \leftarrow j$ s.t. $(\{m\} \cup \mathbf{De}(m)) \cap \mathbf{B} = \emptyset$.

Definition 4. *A path p between T and Y is a T-**backdoor path** if some directed edge on p has T as its destination.*

Definition 5 (Back-Door Criterion [33]). *A set $\mathbf{Z}$ satisfies the **backdoor criterion** relative to variables (T, Y) in a DAG $\mathcal{G}$ if:*

(i) $\mathbf{Z} \cap \mathbf{De}(T) = \emptyset$; and
(ii) $\mathbf{Z}$ blocks every T-backdoor path between T and Y.

In this case, $\mathbf{Z}$ is a sufficient adjustment set for correctly estimating $ATE(T, Y)$ (Definition 1).

6.2 The MapRemoval Algorithm

Given a causal graph $\mathcal{G}$, MapRemoval (Algorithm 5) finds a set of edge edits that prohibits variable V from being an adjustment variable. Intuitively, Algorithm 5 makes V violate first condition of Definition 5, which states that a descendant of T cannot be an adjustment variable. We consider two cases:

Algorithm 6 Suggest causal graph edits that would include a variable in the adjustment set.

```
1:  function MapAddition(G, F, B, V)
2:      edits ← []
3:      if V ∈ De(T) ∨ V ∈ De(Y)
4:          P ← DirPaths(T, V) ∪ DirPaths(Y, V)
5:          for p ∈ P
6:              (V_i → V_j) ← the last Not Fixed edge in p.
7:              if (V_i → V_j) does not exist
8:                  return []
9:              edits += (V_i, V_j, Remove)
10:     edits += (V, T, Add)
11:     edits += (V, Y, Add)
12:     return Deduplicate(edits)
```

Case 1: V is not an ancestor of T. V can be made a descendant of T by adding an edge from a variable $W \in (T \cup \mathbf{De}(T))$ to V without creating directed cycles. We suggest one such edit that minimizes the directed distance from W to T but also respects $\mathbf{B}$ by being *ABSENT* as opposed to *BANNED*.

Case 2: V is an ancestor of T. In this case, directly applying the approach in Case 1 creates a cycle. To fix this, we must first break all directed paths from V to T. For each such path, we identify the closest non-*FIXED* edge to V and remove it to break the path. If all edges on one such path are *FIXED*, Algorithm 5 returns no edits since it is impossible to include V in $\mathbf{De}(T)$ without creating a cycle. We remove duplicate edge removal suggestions from overlapping directed paths from V to T at the end.

Even though we present MapRemoval in Algorithm 5 to facilitate understanding, it can be implemented with time complexity $O(|\mathbf{E}|+|\mathbf{V}|)$. Breaking the directed paths from V to T involves identifying at most $|\mathbf{E}|$ edges to remove; this can be done efficiently by finding the sub-graph that contains only these paths and running a breadth-first search backwards from T. Then, finding the right directed edge to add or flip requires checking at most $|\mathbf{V}|$ possible sources.

6.3 The MapAddition Algorithm

Given a causal graph $\mathcal{G}$ and a treatment-outcome pair (T, Y), MapAddition (Algorithm 6) finds a set of edge edits that makes variable V an obligatory adjustment variable. Intuitively, Algorithm 6 is built on the second condition of Definition 5, which states that any set satisfying the backdoor criterion must block every T-backdoor path. We create a T-backdoor path p that can only be blocked by V. To ensure this, we make V part of a fork along p and adjacent to both T and Y via p, so that p cannot be blocked by some other variable. Similar to MapRemoval, there are two cases:

Case 1. V is not a descendant of T or Y. The first condition of Definition 5 is not violated since $V \not\in \mathbf{De}(T)$. It suffices to create the path described above by adding edges $V \rightarrow T$ and $V \rightarrow Y$, which will not create a cycle since $V \not\in \mathbf{De}(Y)$.

Case 2. V is a descendant of T or Y. First, we break these descendant relations similar to Case 2 of MapRemoval. We identify the closest non-*FIXED* edge to V by iterating backwards on the paths from T/Y to V and remove it to break the path. We handle the edge case where some path cannot be broken, as well as possible duplicate edits, similar to MapRemoval.

MapAddition can also be implemented more efficiently than the version shown in Algorithm 6. In particular, breaking the paths that make V a descendant of T and/or Y in the same manner as discussed earlier for MapRemoval leads to time complexity $O(|\mathbf{E}|)$.

7 EVALUATION

We will now present some early results of the strategies above, in terms of their ability to guide the user towards a higher-confidence causal graph. In this preliminary evaluation, we focus on testing our approaches on a variety of ground-truth causal graphs and associated datasets, while starting from a number of different initial causal graphs. Due to space constraints, we leave a fuller sensitivity study of our approach to problem parameters (e.g. graph sparsity, number of variables) to future work.

7.1 Experimental Setup

7.1.1 Environment. All experiments were run on a machine equipped with two 20-core 2.10 GHz Intel Xeon Gold 6230 CPUs [25] and 256 GiB of memory, running Linux 5.19.12.

7.1.2 Datasets. We generate evaluation datasets in two steps. First, GenDAG(N, p, w_{min}, w_{max}, n_{min}, n_{max}), creates a graph $\mathcal{G}$ by instantiating N nodes in topological order and adding each edge from a node to each successor of it with probability p. Each edge has an associated *weight* drawn uniformly from $[w_{min}, w_{max}]$ and an associated *noise standard deviation*, drawn uniformly from $[n_{min}, n_{max}]$.

Then, GenData($\mathcal{G}$, L, v_{min}, v_{max}) creates a dataset of L points. For each data point, each variable associated with a source node in $\mathcal{G}$ is drawn uniformly from $[v_{min}, v_{max}]$. The rest are assigned values based on their incoming edges. In particular, for each incoming edge e, we multiply the value of the source of e with the weight of e and add Gaussian noise with zero mean and the noise standard deviation of e. The value of a non-source variable is the sum of all the values computed this way for each of its incoming edges.

7.1.3 User Model. ECCS is fundamentally an interactive system. In order to evaluate it experimentally, we simulate a specific user behavior, stylized as User. A User begins invoking ECCS with a starting graph $\mathcal{G}_{init}$ obtained by calling GenDAG and empty sets of *FIXED* and *BANNED* edges, $\mathbf{F}_0 = \emptyset$ and $\mathbf{B}_0 = \emptyset$. The User also specifies a dataset D, a treatment variable T and an outcome variable Y. To simulate a human expert user able to evaluate the suggested edge edits of ECCS, a User also has access to the ground truth graph $\mathcal{G}_{real}$ used to generate D. In each interaction round, a User sequentially considers each edge for which ECCS suggested an edit. For each such edge, a User marks it and its reverse as *FIXED* and/or *BANNED* based on $\mathcal{G}_{real}$, even if that is not the edit type that ECCS suggested. That is, if ECCS suggested (V_i, V_j, *ADD*) but $\mathcal{G}_{real}$ has an edge $V_j \rightarrow V_i$, a User will mark $V_j \rightarrow V_i$ as *FIXED* and $V_i \rightarrow V_j$ as *BANNED*. This simulates an expert verifier whose attention is directed by ECCS to the two edge endpoints V_i, V_j and who can provide a verdict for their causal relationship. As mentioned in Section 6, AdjSetEdit may initiate a new interaction round before the User has produced a judgment for each of the suggested edits. In this case, the User simply continues by producing judgments for the suggested edits in the new round.

Ground Truth Graphs	GenDAG(10, 0.5, −10, 10, 0.001, 2)	10 runs
Datasets	GenData($\mathcal{G}$, 1000, −10, 10)	3 runs/graph
Starting Graphs	GenDAG(10, 0.5, −10, 10, 0.001, 2)	10 runs
Choices of T/Y	Combinations$\mathcal{G}.nodes$, 2	$\binom{10}{2} = 45$
Strategies	RandomEdit	3 runs
	SingleEdit	1 run
	HeuristicEdit	1 run
	AdjSetEdit	1 run
Total	**$10 \times 3 \times 10 \times 45 \times (1 + 1 + 1 + 3) = 81{,}000$**	

Table 1: Breakdown of our experiment instances.

7.1.4 Baseline. As a baseline for our proposed strategies we define RandomEdit, which suggests random single-edge edits. The strategy samples elements of OneStepEdits($\mathcal{G}$, **F**, **B**) uniformly at random without replacement and returns the first sampled element that leads to a graph for which IsAcceptable returns `True`. If no such element exists, it does not return any suggested edits. A User invoking this strategy is guaranteed to arrive at the correct causal graph, and therefore also at the correct ATE, in at most $\binom{|\mathbf{V}|}{2}$ calls, corresponding to the unordered pairs of distinct elements of **V**.

7.1.5 Experiment instances. We ran a total of 81,000 experiment instances, as indicated in Table 1. We created 10 10-node "ground truth" causal graphs and 3 associated 1000-point datasets for each. We then created 10 more 10-node graphs from which the User starts their interaction with ECCS. For each combination of a ground truth graph, a corresponding dataset and a starting graph, we set the user's choice of T and Y to all possible options. Because of the topological order-based approach of GenDAG, we know that a directed path from V_i to V_j cannot exist if $j < i$, so we prune the corresponding pairs and get 45 valid pairs. Some experiment instances are deemed invalid because there is no directed path from the selected treatment to the selected outcome in both the ground truth DAG and in the starting graph. The results we present below are aggregated over *valid* experiment instances. Finally, for each combination of (ground truth graph, starting graph, T, Y), we simulate each of our three strategies for 10 judgments. For HeuristicEdit, we set $budget = 1000$, $m = 2, k = 100$. For the randomized baseline, we repeat such a 10-judgment interaction 3 times and average the performance. Note that certain strategies may stop producing graph edit suggestions before the 10^{th} judgment is made, for example if all the suggestions they could have produced are not acceptable. In such cases, we assume that the metrics of interest remain constant going forward.

7.1.6 Metrics. For each strategy and each experiment instance, we will solve an instance of the Interactive Causal Graph Verification problem (see Section 3.1) with $J = 10$. Instead of enforcing a hard upper limit on L, we will monitor the latency of ECCS in each interaction round, averaged across all the experiment instances for which that round was reached. At the same time, we will monitor the evolution of two metrics over the 10 User judgments (again averaged over experimental instances):

- **Absolute Relative Error (ARE) [4] in ATE**, defined as $ARE\_ATE = \frac{ATE_{current} - ATE_{ground\_truth}}{ATE_{ground\_truth}}$. This is the primary metric we are interested in.
- **Graph Edit Distance** between the graph produced by the User after the most recent judgment, and the ground truth causal graph. Although this metric is not directly related to **Problem 1**, it can help give some extra insight. The distance is incremented by one for each edge edit needed to transform one graph to the other, where the allowed operations are *ADD*, *REMOVE* and *FLIP*.

7.2 RandomEdit

We will first briefly discuss the performance of RandomEdit. On the ARE_ATE, Figure 2a reveals that the impact of this strategy is negligible and non-monotonic, as there is no connection between the suggested edits and the ATE of interest. After 10 judgments, this strategy achieves an average ARE_ATE of 0.42. Figure 2b reveals that this strategy decreases graph edit distance roughly linearly on average, as expected from the User's reactions to randomized suggested graph edits, reaching an average edit distance of 17.23 after 10 judgments. However, as evident from Figure 2c, this approach is fast, with each call into ECCS taking an average of 0.16 seconds.

7.3 SingleEdit

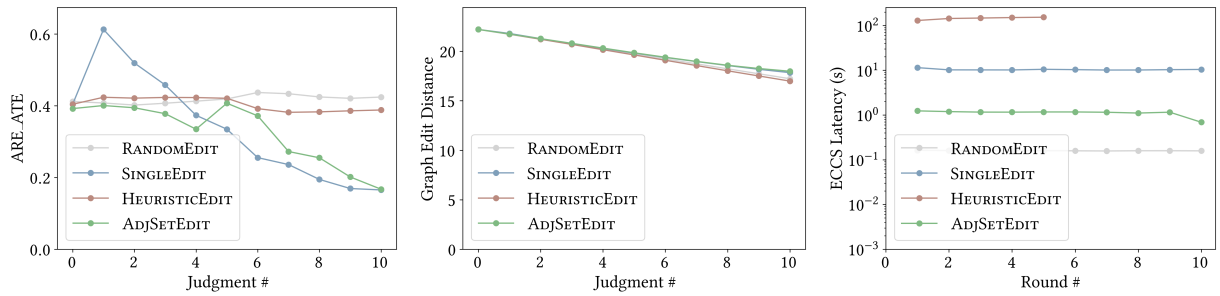
Moving on to SingleEdit, per Figure 2a, the impact on the primary metric of ARE_ATE is dramatic, reaching 0.17 after 10 judgments - a reduction of 61.06% compared to RandomEdit. At the same time, Figure 2b reveals that the decrease in graph edit distance is only negligibly smaller than RandomEdit, reaching 17.84 on average after 10 judgments. As shown in Figure 2c, the price for the ARE_ATE improvement is a latency increase by 2 orders of magnitude, with each call to ECCS requiring on average 10.39 seconds when using this strategy.

7.4 HeuristicEdit

Despite its increased sophistication, HeuristicEdit did not offer meaningful improvements on the metrics of interest. Figure 2a shows that it achieves an average ARE_ATE of 0.39 after 10 judgments without a clear decreasing trend, an improvement of only 8.51% over RandomEdit. Its impact on graph edit distance nevertheless exceeds that of other strategies, reaching a final average value of 16.98. Still, with an average latency of 144.90 seconds per interaction round, further work is required to make the ideas of HeuristicEdit useful. The main drawback of this method (beyond the long latency imposed by the heuristic search) appears to be its sequential identification of *different* suggested edge edits that attempt to achieve the inclusion or omission of the *same* impactful adjustment variable, even when the User keeps rejecting them.

7.5 AdjSetEdit

Finally, turning to AdjSetEdit, we observe from Figure 2a that performance on the primary metric of interest (ARE_ATE) is comparable to SingleEdit: after 10 judgments, it achieves a value of 0.17, a reduction of 60.60% compared to the baseline. The trajectory of the graph edit distance also mirrors that of SingleEdit, reaching 17.99 on average after 10 judgments. However, this strategy offers a 9.28× improvement in latency compared to SingleEdit, with Figure 2c showing that each call to ECCS only takes 1.12 seconds on average. This latency improvement is substantial because it brings the system within an "interactive" operating domain, something that SingleEdit and HeuristicEdit do not offer.



(a) ARE_ATE **(b) Graph Edit Distance** **(c) Latency**

Figure 2: The evolution of our metrics of interest for each strategy over 10 User judgments.

8 RELATED WORK

Causality and Data Management A wide variety of data management tasks have benefited from advances in the theory of causality, including query result and classification explanation [5, 38], hypothetical reasoning [11], exploratory data analysis [29], fault localization [3, 12, 31], data integration [42, 59], model fairness [62] and maximizing RCT data usefulness [6]. Any such system can benefit from the ability of ECCS to improve the quality of the causal graph.

General Causal Discovery Causal discovery has been an active research area for decades [14, 28, 43, 47, 48, 53, 56, 60, 63]. Some existing algorithms are "constraint-based" [47, 48], while others try to estimate the parameters of functional causal models [43]. A subset of existing algorithms focus specifically on *local* causal discovery around the treatment variable, in order to reduce computational cost and characterize the range of possible causal effect sizes based on observational data [15, 30, 55, 58]. Yet another class of algorithms rely on the variable *names* [17, 18, 20] to deduce causal relations, most recently using large language models [26, 59]. Despite the success of such algorithms, they each have failure modes related to the assumptions they make about the data or about the meaningfulness of variable names [14, 26]. ECCS aims to efficiently use expert user feedback to tackle such failure modes.

Data-driven Causal Graph Refinement One approach to causal graph refinement has been targeted data collection through judicious system interventions [10, 22, 23, 27, 35, 41]. This approach may be necessary when expert verification is unsuited for determining the direction of causality. In contrast, ECCS targets domains where such verification is possible, so that intervening is not necessary. Moreover, these works often build on a computed skeleton/the Markov Equivalence Class [7, 19, 46] and focus on determining the direction of undirected edges in the skeleton, which limits the possibility of refining over the causal structure itself, unlike ECCS. Another line of work combats the possibility of spurious correlations being identified as causal relationships, by setting out to find a class of necessary and sufficient causal graphs [8] over only a relevant subset of variables. While this approach can lead to a better-quality causal graph after the initial automatic causal discovery step, it is complementary to invoking ECCS on the resulting graph to also incorporate expert knowledge about the domain.

Adjustment Sets and ATE Calculation Under the directed acyclic graph model, we use Pearl's backdoor criterion [33] to determine a valid adjustment set. There exist variants of the backdoor criterion [52]. Another line of research around identifying a causal model and the ATE works in the domain of Markov Equivalence Classes and completed partially directed acyclic graphs (CPDAGs) [7, 19, 50]. A notable recent work [15] uses the observation that each possible valid value of the ATE corresponds to a valid set of parents of the treatment to adjust for in the CPDAG. It uses local constraint-based causal discovery to efficiently identify the Markov Blanket of the treatment, all possible valid parent sets to adjust for, and therefore all possible ATE values. This algorithm still has exponential complexity in bad cases. However, unlike our work, [15] does not aim to find ATE values that are close to ground truth. Previous work has also examined this problem in ancestral graphs [51].

9 CONCLUSION AND FUTURE WORK

In this paper, we approached the incorporation of human knowledge into automatically-generated causal graphs by formulating the Interactive Causal Graph Verification problem and exploring three early algorithmic strategies for addressing it. We also presented a preliminary evaluation of these strategies on synthetic workloads. We hope to solicit community feedback on our problem formulation, as well as exchange ideas on the best ways to tackle it.

We are in the process of diving deeper into this problem and developing solutions that provide both tighter optimality guarantees and better runtime performance. A long-standing main barrier for non-naive algorithms comes from the complexity of statistically scoring causal graphs and/or subgraphs, either in the more canonical PC-like algorithms or more recent works such as [35]. Combining the local computation of these scoring methods with our framework of exploration and local edits may be promising. Additionally leveraging language information in the names of causal variables may yield more cost-effective graph scoring schemes.

On the theoretical side, we are interested in the informational theoretical complexity of this problem; namely, the number of causal graph evaluations needed to estimate the edges with the highest impact for a given causal question to a certain accuracy. We are also interested in combining more observations from the adjustment set identification literature with our framework.

ACKNOWLEDGMENTS

We gratefully acknowledge the support of the DARPA ASKEM project (Award No. HR00112220042).

REFERENCES

[1] Emile HL Aarts, Jan HM Korst, and Peter JM van Laarhoven. 1988. A Quantitative Analysis of the Simulated Annealing Algorithm: A Case Study for the Traveling Salesman Problem. *Journal of Statistical Physics* 50 (1988), 187–206.

[2] Nadia Abd-Alsabour. 2014. A Review on Evolutionary Feature Selection. In *2014 European Modelling Symposium*. IEEE Computer Society CPS, Los Alamitos, CA, USA, 20–26. https://doi.org/10.1109/EMS.2014.28

[3] Pooja Aggarwal, Ajay Gupta, Prateeti Mohapatra, Seema Nagar, Atri Mandal, Qing Wang, and Amit Paradkar. 2021. Localization of Operational Faults in Cloud Applications by Mining Causal Dependencies in Logs Using Golden Signals. In *Service-Oriented Computing – ICSOC 2020 Workshops*. Springer International Publishing, New York, NY, USA, 137–149.

[4] OECD AI. 2024. Absolute Relative Error (ARE). Retrieved May 29, 2024 from https://oecd.ai/en/catalogue/metrics/absolute-relative-error-%28are%29

[5] Kamran Alipour, Aditya Lahiri, Ehsan Adeli, Babak Salimi, and Michael Pazzani. 2022. Explaining Image Classifiers Using Contrastive Counterfactuals in Generative Latent Spaces. arXiv:2206.05257 [cs.CV]

[6] Abdullah Alomar, Pouya Hamadanian, Arash Nasr-Esfahany, Anish Agarwal, Mohammad Alizadeh, and Devavrat Shah. 2023. CausalSim: A Causal Framework for Unbiased Trace-Driven Simulation. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. USENIX Association, Berkeley, CA, USA, 1115–1147.

[7] Steen A. Andersson, David Madigan, and Michael D. Perlman. 1997. A Characterization of Markov Equivalence Classes for Acyclic Digraphs. *The Annals of Statistics* 25, 2 (1997), 505 – 541. https://doi.org/10.1214/aos/1031833662

[8] Hengrui Cai, Yixin Wang, Michael Jordan, and Rui Song. 2023. On Learning Necessary and Sufficient Causal Graphs. In *Advances in Neural Information Processing Systems*, Vol. 36. Curran Associates, Inc., Red Hook, NY, USA, 42148–42160. https://proceedings.neurips.cc/paper_files/paper/2023/file/837b396039248acb08c385bebb6291b4-Paper-Conference.pdf

[9] Leonid Chindelevitch, Daniel Ziemek, Ahmed Enayetallah, Ranjit Randhawa, Ben Sidders, Christoph Brockel, and Enoch S. Huang. 2012. Causal reasoning on biological networks: interpreting transcriptional changes. *Bioinformatics* 28, 8 (02 2012), 1114–1121. https://doi.org/10.1093/bioinformatics/bts090 arXiv:https://academic.oup.com/bioinformatics/article-pdf/28/8/1114/48930575/bioinformatics_28_8_1114.pdf

[10] Davin Choo, Kirankumar Shiragur, and Arnab Bhattacharyya. 2022. Verification and search algorithms for causal DAGs. In *Advances in Neural Information Processing Systems*, Vol. 35. Curran Associates, Inc., Red Hook, NY, USA, 12787–12799. https://proceedings.neurips.cc/paper_files/paper/2022/file/5340b0c0b76dc0115f5cc91c20c1251d-Paper-Conference.pdf

[11] Sainyam Galhotra, Amir Gilad, Sudeepa Roy, and Babak Salimi. 2022. HypeR: Hypothetical Reasoning With What-If and How-To Queries Using a Probabilistic Causal Approach. In *Proceedings of the 2022 International Conference on Management of Data* (Philadelphia, PA, USA) *(SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 1598–1611. https://doi.org/10.1145/3514221.3526149

[12] Yu Gan, Mingyu Liang, Sundar Dev, David Lo, and Christina Delimitrou. 2021. Sage: Practical and Scalable ML-Driven Performance Debugging in Microservices. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (Virtual, USA) *(ASPLOS '21)*. Association for Computing Machinery, New York, NY, USA, 135–151. https://doi.org/10.1145/3445814.3446700

[13] Maxime Gasse, Damien Grasset, Guillaume Gaudron, and Pierre-Yves Oudeyer. 2021. Causal Reinforcement Learning using Observational and Interventional Data. arXiv:2106.14421 [cs.LG]

[14] Clark Glymour, Kun Zhang, and Peter Spirtes. 2019. Review of Causal Discovery Methods Based on Graphical Models. *Frontiers in Genetics* 10 (2019), 524.

[15] Shantanu Gupta, David Childers, and Zachary Chase Lipton. 2023. Local Causal Discovery for Estimating Causal Effects. In *Proceedings of the Second Conference on Causal Learning and Reasoning (Proceedings of Machine Learning Research, Vol. 213)*. PMLR, 408–447. https://proceedings.mlr.press/v213/gupta23b.html

[16] Peter E Hart, Nils J Nilsson, and Bertram Raphael. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE transactions on Systems Science and Cybernetics* 4, 2 (1968), 100–107.

[17] Chikara Hashimoto. 2019. Weakly Supervised Multilingual Causality Extraction from Wikipedia. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 2988–2999. https://doi.org/10.18653/v1/D19-1296

[18] Oktie Hassanzadeh, Debarun Bhattacharjya, Mark Feblowitz, Kavitha Srinivas, Michael Perrone, Shirin Sohrabi, and Michael Katz. 2019. Answering Binary Causal Questions Through Large-Scale Text Mining: An Evaluation Using Cause-Effect Pairs from Human Experts.. In *IJCAI*. 5003–5009.

[19] Alain Hauser and Peter Bühlmann. 2012. Characterization and Greedy Learning of Interventional Markov Equivalence Classes of Directed Acyclic Graphs. *Journal of Machine Learning Research* 13, 1 (Aug 2012), 2409–2464.

[20] Stefan Heindorf, Yan Scholten, Henning Wachsmuth, Axel-Cyrille Ngonga Ngomo, and Martin Potthast. 2020. CauseNet: Towards a Causality Graph Extracted from the Web. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (Virtual Event, Ireland) *(CIKM '20)*. Association for Computing Machinery, New York, NY, USA, 3023–3030. https://doi.org/10.1145/3340531.3412763

[21] Chanelle J Howe, Zinzi D Bailey, Julia R Raifman, and John W Jackson. 2022. Recommendations for Using Causal Diagrams to Study Racial Health Disparities. *Am J Epidemiol* 191, 12 (Nov. 2022), 1981–1989.

[22] Huining Hu, Zhentao Li, and Adrian R Vetta. 2014. Randomized Experimental Design for Causal Graph Discovery. In *Advances in Neural Information Processing Systems*, Vol. 27. Curran Associates, Inc., Red Hook, NY, USA. https://proceedings.neurips.cc/paper_files/paper/2014/file/e53a0a2978c28872a4505bdb51db06dc-Paper.pdf

[23] Antti Hyttinen, Frederick Eberhardt, and Patrik O Hoyer. 2013. Experiment Selection for Causal Discovery. *Journal of Machine Learning Research* 14, 1 (2013), 3041–3071.

[24] Guido W. Imbens. 2020. Potential Outcome and Directed Acyclic Graph Approaches to Causality: Relevance for Empirical Practice in Economics. *Journal of Economic Literature* 58, 4 (December 2020), 1129–79. https://doi.org/10.1257/jel.20191597

[25] Intel Corporation. 2019. *Intel Xeon Gold 6230 CPU*. Intel Corporation. Retrieved May 29, 2024 from https://ark.intel.com/content/www/us/en/ark/products/192437/intel-xeon-gold-6230-processor-27-5m-cache-2-10-ghz.html

[26] Emre Kıcıman, Robert Ness, Amit Sharma, and Chenhao Tan. 2023. Causal Reasoning and Large Language Models: Opening a New Frontier for Causality. arXiv:2305.00050 [cs.AI]

[27] Murat Kocaoglu, Alex Dimakis, and Sriram Vishwanath. 2017. Cost-Optimal Learning of Causal Graphs. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*. PMLR, 1875–1884. https://proceedings.mlr.press/v70/kocaoglu17a.html

[28] Sindy Löwe, David Madras, Richard Zemel, and Max Welling. 2022. Amortized Causal Discovery: Learning to Infer Causal Graphs from Time-Series Data. In *Proceedings of the First Conference on Causal Learning and Reasoning (Proceedings of Machine Learning Research, Vol. 177)*. PMLR, 509–525. https://proceedings.mlr.press/v177/lowe22a.html

[29] Pingchuan Ma, Rui Ding, Shuai Wang, Shi Han, and Dongmei Zhang. 2023. XInsight: eXplainable Data Analysis Through The Lens of Causality. *Proceedings of the ACM on Management of Data* 1, 2 (2023), 1–27.

[30] Marloes H. Maathuis, Markus Kalisch, and Peter Bühlmann. 2009. Estimating High-dimensional Intervention Effects from Observational Data. *The Annals of Statistics* 37, 6A (2009), 3133 – 3164. https://doi.org/10.1214/09-AOS685

[31] Markos Markakis, An Bo Chen, Brit Youngmann, Trinity Gao, Ziyu Zhang, Rana Shahout, Peter Baile Chen, Chunwei Liu, Ibrahim Sabek, and Michael Cafarella. 2024. Sawmill: From Logs to Causal Diagnosis of Large Systems. In *Companion of the 2024 International Conference on Management of Data (SIGMOD-Companion '24), June 9–15, 2024, Santiago, AA, Chile*. Association for Computing Machinery, New York, NY, USA, 4 pages. https://doi.org/10.1145/3626246.3654731

[32] Judea Pearl. 1985. Bayesian Networks: A Model of Self-activated Memory for Evidential Reasoning. In *Proceedings of the 7th Annual Conference of the Cognitive Science Society*. 329–334.

[33] Judea Pearl. 2009. *Causality: Models, Reasoning and Inference*. Cambridge University Press, New York, NY, USA.

[34] Judea Pearl and Dana Mackenzie. 2018. *The Book of Why: The New Science of Cause and Effect*. Basic Books, New York, NY, USA.

[35] Chen Peng, Di Zhang, and Urbashi Mitra. 2024. Graph Identification and Upper Confidence Evaluation for Causal Bandits with Linear Models. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 7165–7169. https://doi.org/10.1109/ICASSP48485.2024.10445823

[36] Vitalii I Rodionov. 1992. On the number of labeled acyclic digraphs. *Discrete Mathematics* 105, 1-3 (1992), 319–321.

[37] Karen Sachs, Omar Perez, Dana Pe'er, Douglas A. Lauffenburger, and Garry P. Nolan. 2005. Causal Protein-Signaling Networks Derived from Multiparameter Single-Cell Data. *Science* 308, 5721 (2005), 523–529. https://doi.org/10.1126/science.1105809 arXiv:https://www.science.org/doi/pdf/10.1126/science.1105809

[38] Babak Salimi, Johannes Gehrke, and Dan Suciu. 2018. Bias in OLAP Queries: Detection, Explanation, and Removal. In *Proceedings of the 2018 International Conference on Management of Data* (Houston, TX, USA) *(SIGMOD '18)*. Association for Computing Machinery, New York, NY, USA, 1021–1035. https://doi.org/10.1145/3183713.3196914

[39] Pedro Sanchez, Jeremy P Voisey, Tian Xia, Hannah I Watson, Alison Q O'Neil, and Sotirios A Tsaftaris. 2022. Causal machine learning for healthcare and precision medicine. *Royal Society Open Science* 9, 8 (2022), 220638.

[40] Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. 2021. Towards Causal Representation Learning. arXiv:2102.11107 [cs.LG]

[41] Karthikeyan Shanmugam, Murat Kocaoglu, Alexandros G Dimakis, and Sriram Vishwanath. 2015. Learning Causal Graphs with Small Interventions. In *Advances*

in Neural Information Processing Systems, Vol. 28. Curran Associates, Inc., Red Hook, NY, USA. https://proceedings.neurips.cc/paper_files/paper/2015/file/b865367fc4c0845c0682bd466e6ebf4c-Paper.pdf

[42] Xu Shi, Ziyang Pan, and Wang Miao. 2023. Data integration in causal inference. *Wiley Interdisciplinary Reviews: Computational Statistics* 15, 1 (2023), e1581.

[43] Shohei Shimizu, Patrik O. Hoyer, Aapo Hyvärinen, Antti Kerminen, and Michael Jordan. 2006. A Linear Non-Gaussian Acyclic Model for Causal Discovery. *Journal of Machine Learning Research* 7, 10 (2006), 2003 – 2030. https://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=23240079&site=eds-live&scope=site

[44] N. J. A. Sloane. [n. d.]. A003024: Number of acyclic digraphs (or DAGs) with n labeled nodes. Retrieved May 29, 2024 from https://oeis.org/A003024

[45] Michael E. Sobel. 2000. Causal Inference in the Social Sciences. *J. Amer. Statist. Assoc.* 95, 450 (01 Jun 2000), 647–651. https://doi.org/10.1080/01621459.2000.10474243

[46] Arjun Sondhi and Ali Shojaie. 2019. The Reduced PC-Algorithm: Improved Causal Structure Learning in Large Random Networks. *Journal of Machine Learning Research* 20, 164 (2019), 1–31. http://jmlr.org/papers/v20/17-601.html

[47] Peter Spirtes and Clark Glymour. 1991. An Algorithm for Fast Recovery of Sparse Causal Graphs. *Social Science Computer Review* 9, 1 (1991), 62–72.

[48] Peter Spirtes, Clark N Glymour, and Richard Scheines. 2000. *Causation, Prediction, and Search*. MIT Press, Cambridge, MA, USA.

[49] Jin Tian, Azaria Paz, and Judea Pearl. 1998. *Finding Minimal D-separators*.

[50] Benito van der Zander and Maciej Liskiewicz. 2016. Separators and Adjustment Sets in Markov Equivalent DAGs. *Proceedings of the AAAI Conference on Artificial Intelligence* 30, 1 (Mar. 2016). https://doi.org/10.1609/aaai.v30i1.10424

[51] Benito van der Zander, Maciej Liśkiewicz, and Johannes Textor. 2014. Constructing Separators and Adjustment Sets in Ancestral Graphs. In *Proceedings of the UAI 2014 Conference on Causal Inference: Learning and Prediction - Volume 1274* (Quebec City, Canada) *(CI'14)*. CEUR-WS.org, Aachen, DEU, 11–24.

[52] Benito van der Zander, Maciej Liśkiewicz, and Johannes Textor. 2019. Separators and adjustment sets in causal graphs: Complete criteria and an algorithmic framework. *Artificial Intelligence* 270 (2019), 1–40. https://doi.org/10.1016/j.artint.2018.12.006

[53] Thomas Verma and Judea Pearl. 1990. Equivalence and Synthesis of Causal Models. In *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence (UAI '90)*. Elsevier Science Inc., USA, 255–270.

[54] Matthew J. Vowels, Necati Cihan Camgoz, and Richard Bowden. 2022. D'ya Like DAGs? A Survey on Structure Learning and Causal Discovery. *ACM Comput. Surv.* 55, 4, Article 82 (Nov 2022), 36 pages. https://doi.org/10.1145/3527154

[55] Changzhang Wang, You Zhou, Qiang Zhao, and Zhi Geng. 2014. Discovering and orienting the edges connected to a target variable in a DAG via a sequential local learning approach. *Computational statistics & data analysis* 77 (2014), 252–266.

[56] Marco A Wiering et al. 2002. Evolving causal neural networks. In *Proceedings of the Twelfth Belgian-Dutch Conference on Machine Learning*. 103–108.

[57] Thomas C. Williams, Cathrine C. Bach, Niels B. Matthiesen, Tine B. Henriksen, and Luigi Gagliardi. 2018. Directed acyclic graphs: a tool for causal studies in paediatrics. *Pediatric Research* 84, 4 (01 Oct 2018), 487–493. https://doi.org/10.1038/s41390-018-0071-3

[58] Jianxin Yin, You Zhou, Changzhang Wang, Ping He, Cheng Zheng, and Zhi Geng. 2008. Partial orientation and local structural learning of causal networks for prediction. In *Proceedings of the Workshop on the Causation and Prediction Challenge at WCCI 2008 (Proceedings of Machine Learning Research, Vol. 3)*. PMLR, 93–105. http://proceedings.mlr.press/v3/yin08a.html

[59] Brit Youngmann, Michael Cafarella, Babak Salimi, and Anna Zeng. 2023. Causal Data Integration. *Proc. VLDB Endow.* 16, 10 (jun 2023), 2659–2665. https://doi.org/10.14778/3603581.3603602

[60] Jiaming Zeng, Michael F Gensheimer, Daniel L Rubin, Susan Athey, and Ross D Shachter. 2022. Uncovering interpretable potential confounders in electronic medical records. *Nature Communications* 13, 1 (2022), 1014.

[61] Yan Zeng, Ruichu Cai, Fuchun Sun, Libo Huang, and Zhifeng Hao. 2023. A Survey on Causal Reinforcement Learning. arXiv:2302.05209 [cs.AI]

[62] Jiongli Zhu, Sainyam Galhotra, Nazanin Sabri, and Babak Salimi. 2023. Consistent Range Approximation for Fair Predictive Modeling. arXiv:2212.10839 [cs.LG]

[63] Shengyu Zhu, Ignavier Ng, and Zhitang Chen. 2020. Causal Discovery with Reinforcement Learning. arXiv:1906.04477 [cs.LG]