A Framework for Practical Causal Discovery from Observational Data

Gustavo F.V. de Oliveira[1], Fabrício A. Silva[1], and Marcus H.S. Mendes[1]

IEF, Universidade Federal de Viçosa, Florestal, Brazil
{gustavo.viegas,fabricio.asilva,marcus.mendes}@ufv.br

Abstract. Causal reasoning is crucial for advancing AI models beyond correlation-based predictions to infer meaningful cause-and-effect relationships. This paper introduces a novel framework that facilitates causal inference for AI developers with minimal causality expertise. It integrates multiple causal discovery algorithms, standardizes data structures, and employs parallel processing to generate and evaluate causal models efficiently. It automates the selection of appropriate algorithms based on data attributes and prioritizes feature importance to ensure robust causal effect estimation and refutation. The framework effectiveness is demonstrated through a benchmark on a real-world dataset, where it accurately identifies known causal relationships. By providing user-friendly outputs and customizable features, the framework enhances the accessibility and reliability of causal analysis in AI, paving the way for its application in sensitive domains such as healthcare, finance, and justice.

Keywords: Causality · Causal Discovery · Framework · Ethical AI.

1 Introduction

Causation is the concept referring to the cause-and-effect relationship wherein one event (the cause) leads to another event (the effect) and establishes a direct connection or influence between them [17]. Artificial Intelligence (AI) models have demonstrated remarkable capabilities in establishing intricate connections and making accurate predictions from data. However, when confronted with unfamiliar data, these models face challenges, leading to inaccurate causal relationships [16]. This limitation is particularly troubling in ethical AI, where biased associations are highly undesirable, such as the risk of incorrect associations between factors like race and criminal behavior, resulting in unjust outcomes [6].

The complexity of the field, with its requirements of domain knowledge, causal Directed Acyclic Graph (DAG) notation, causal discovery algorithms, estimators, and refuters, can be discouraging. While the adoption of causality analysis has the potential to enhance AI models, the incorporation of these concepts can be overwhelming and challenging due to the steep learning curve and the need to stay current with evolving content. The early stage of development of causal frameworks poses a challenge to their widespread adoption. The field is relatively new, with the most popular and corporate-backed packages released within the last five years [21,24,5]. Additionally, essential performance features like time limits and parallelism still need to be universally implemented, which could be a significant burden for certain self-learning AI models to overcome.

To facilitate the adoption of causal inference, this paper introduces *Causal-Nest*, a user-friendly framework simplifying causal inference for AI developers with limited causality knowledge. The primary goal of the framework is to build multiple causal models using relevant causal discovery algorithms while considering time constraints and prioritizing potential causal models and their relationships. It can automatically select suitable causal discovery algorithms based on the input data and standardize input and output formats across all processes. Additionally, it produces a graphical output summarizing the pipeline results in a format that is easy for humans to understand.

The article is organized as follows: Section 2 overviews the main causality pipeline: causal discovery and inference. Section 3 presents our proposed framework. Section 4 discusses experimental results, and Section 5 concludes the work.

2 Related Work

This section discusses key components of a causality pipeline that systematically identifies causal relationships from observational data. The main steps of the pipeline are causal discovery to uncover the causal graph and causal inference to estimate effects. To our knowledge, existing literature lacks comprehensive frameworks that can independently address all aspects of a causality problem without the need for manual algorithm selection, model prioritization, and parallel execution, which would typically necessitate a substantial understanding of causality from the user.

2.1 Causal Discovery (CD)

The process of extracting causality knowledge from data involves key steps. It begins with problem formulation and constructing a causal model using a Directed Acyclic Graph (DAG) to represent hypothesized causal pathways among identified variables. Causal Discovery (CD) refers to the process that creates these models from observational data, and various methodologies exist for uncovering causal graph structures.

Causal discovery methods often yield DAGs or undirected Bayesian networks, each presenting its computational complexities. Several factors must be considered when choosing a causal discovery algorithm, including computational complexity, data assumptions, and the ability to extract meaningful causal relationships [17]. The selection criteria involve analyzing the data types, distribution assumptions, and linearity assumptions [23].

2.2 Causal Inference (CI)

Causal Inference (CI) is a complex task involving estimating causal effects based on a causal graph constructed by domain experts or inferred by advanced causal discovery algorithms. Statistical estimators quantify causal effects between variables in a causal model, and different interpretations of these estimations require domain-specific expertise for accurate understanding [17].

Validating the accuracy of causal effect estimations involves a comprehensive understanding of the data domain and conducting refutation tests. These tests include sensitivity analyses, such as introducing random placebo variables, adding random standard cause variables, or removing subsets from the data to ensure the inferred causal relationships are robust and align with theoretical expectations.

3 *Causal-Nest*

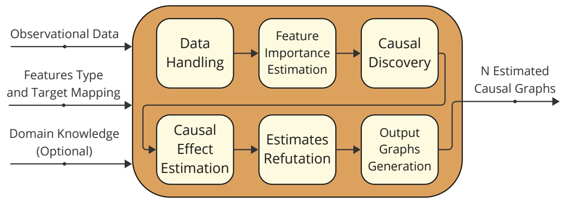


Fig. 1: Summarized pipeline of the proposed framework

This paper proposes a novel framework named *Causal-Nest*[1]. This framework includes a user-friendly interface for easier adoption and automatically manages the many rules and restrictions present in the field of causality. Figure 1 illustrates the key steps in the framework pipeline and their inputs and outputs. All algorithms in the framework were developed in Python 3.9 using packages for graph manipulation, dataset formatting, as well as causal discovery implementation with R backends, and estimation (through high-level packages [9,28,21]). Parallelism was employed in causal discovery, estimation, and refutation using the Pebble package version 5.0.

The framework has a standardized approach to data structures, inputs, and outputs. All graphs are represented as *Networkx* [8] graphs that can be easily translated to DOT, adjacency matrix, and other formats. This standardization ensures a consistent causal process, as the implemented discovery methods in the current causal frameworks operate uniformly while enabling easy customization and adding new methods. The graphical output is in DOT format and can be rendered in SVG, PNG, or other formats.

[1] The code and additional benchmarks are available on GitHub.

3.1 Input and Data Handling

To run the entire pipeline, only a few inputs are necessary: the observational dataset for analysis, the type mapping of variables in the dataset, such as categorical, continuous, and discrete, and the target column representing the observed effect. *Causal-Nest* assumes the input data is processed, cleaned, and formatted. It can handle diverse problem domains and data dimensions. If data is missing, the framework can drop it, apply forward fill or interpolate based on user preference.

Although establishing causality can be challenging, some relationships, such as holidays causing rain, are known to be false upfront. The domain knowledge of forbidden and required edges can be provided as input, formatted in a text file respecting the syntax described in [28].

3.2 Feature Importance Estimation

While correlation does not directly indicate causation, it is improbable for a causal relationship to exist between two variables that are not closely connected. *Causal-Nest* includes user-defined strict time limits for tasks such as effect estimation and refutation. These tasks are ranked in a prioritized list based on their calculated feature importance, which is determined using a Random Forest classifier or regression, and automatically chosen based on the nature of the problem. Establishing this prioritized list helps prevent potentially valuable candidates for causal effects from being overlooked in later stages of the process due to the time limit constraints and their lower position in the sequentially processed list of variables and the time limit constraints.

3.3 Causal Discovery (CD)

Several causal discovery algorithms are available in the literature, each with requirements and restrictions. Choosing the most appropriate one for a particular problem is a complex task. *Causal-Nest* includes twelve CD algorithms and automatically evaluates the ones suitable for the given problem scope, considering the data types of variables, normality distribution (as some models assume Gaussian data), and linearity. Table 1 displays the implemented CD algorithms in *Causal-Nest* with their corresponding backend and type of discovery. The framework also simplifies the process of implementing other discovery methods, both in terms of execution and constraints, to be verified for running in a given dataset. It can be achieved by implementing a straightforward interface that must adhere to two functions: a Boolean function determining if the algorithm is suitable for a given dataset input and another function to output the discovered graph.

The framework validates the data type of each feature in the input dataset to ensure compatibility with the model-accepted data types. If any feature has an unsupported type, the discovery algorithm will be rejected and not applied. A D'Agostino and Pearson-based test is employed for normality checks. This test assumes normality if all features yield a p-value lower than the default threshold of 0.05, applying the null hypothesis that the feature values come from a normal distribution. Lastly, linearity between each variable and the target is assessed using linear regression, expecting the mean of residuals to be lower than the default threshold of 0.05.

CD Algorithm	**Type**	**Backend**
1. Grow-Shrink (GS) [13]	Constraint-based	bnlearn [20]
2. Incremental Association Markov Blanket (IAMB) [25]	Constraint-based	bnlearn [20]
3. Causal Additive Models (CAM) [4]	Constraint-based	CAM R [4]
4. Concave Penalized Coordinate Descent with Reparametrization (CCDr) [3]	Score-based	sparsebn [2]
5. Causal Generative Neural Networks (CGNN) [7]	Score-based	pytorch [1]
6. Greedy Equivalance Search (GES) [15]	Score-based	pcalg [14]
7. Greedy Interventional Equivalance Search (GIES) [15]	Score-based	pcalg [14]
8. Linear Non-Gaussian Acyclic Model (LiNGAM) [22]	Score-based	pcalg [14]
9. Peter-Clark (PC) [11]	Constraint-based	pcalg [14]
10. Structural Agnostic Model (SAM) [10]	Score-based	pytorch [1]
11. Greedy Relaxation of the Sparsest Permutation (GRaSP) [12]	Permutation-based	causallearn [28]
12. BIC Exact Search (BES) [27]	Score-based	causallearn [28]

Table 1: Causal Discovery algorithms implemented by *Causal-Nest*

The framework uses a diverse range of causal discovery algorithms to construct a causal graph from the provided data. This process is parallelized and allows the user to specify the maximum number of workers, which defaults to the number of CPUs in the runtime system. The framework does not prioritize the execution order of algorithms, as the choice depends vastly on the problem domain context. Each worker is responsible for creating the causal graph, generating essential statistics and rankings, and has a user-defined timeout for execution. The automatic algorithm selection, discovery parallelization, timeout execution limits, and user-friendly statistics are fundamental features of the proposed framework, enabling systems that generate real-time data to extract causal knowledge from the data.

Some methods used to derive a causality model from data were not initially designed for causality but for Bayesian networks or other use cases. As a result, the outcome of the discovery algorithms may not always produce a Directed Acyclic Graph (DAG). In such cases, *Causal-Nest* incorporates a custom-defined function to handle these situations. Therefore, every causal discovery graph generated in *Causal-Nest* is a DAG.

The generated statistics in the causal discovery step include the Forbidden Edges Violation Rate (FEVR), the Required Edges Compliance Rate (RECR), and the Knowledge Integrity Score (KIS), which are also contributions of this work. FEVR represents the percentage of forbidden edges present in the generated graph, while RECR indicates the percentage of required edges present in the graph. KIS, a score ranging from 0 to 1, is calculated based on Equation 1, considering FEVR and RECR. A score of 1 indicates that the graph fully adheres to the knowledge restrictions, while a score of 0 indicates that the graph

Algorithm 1: Causal Discovery generated graph priority score

```
   Input  : Graph $G$, target node $t$
   Output: Ranking score
1  if $t \notin G.nodes()$ then return 0
2  $num\_incoming\_edges \leftarrow G$.in_degree($t$)
3  $num\_nodes \leftarrow |G$.nodes()$|$
4  $edge\_score \leftarrow \frac{num\_incoming\_edges}{num\_nodes}$
5  $distances \leftarrow G$.get_nodes_shortest_distance_to(target=$t$)
6  $avg\_distance \leftarrow \frac{\sum_{d \in distances} d}{|distances|}$
7  $distance\_score \leftarrow \frac{1}{avg\_distance+1}$
8  $density\_score \leftarrow G$.density()
9  $connectivity\_score \leftarrow$ 1 if $G$.is_weakly_connected() else 0
10 $bc \leftarrow G$.betweenness_centrality_values()
11 $betweenness\_score \leftarrow \sum_{b \in bc} b$
12 $score \leftarrow 100 \times (edge\_score \times distance\_score \times density\_score \times$
     $connectivity\_score \times betweenness\_score)$
13 return $score$
```

does not adhere to any imposed restrictions. If no knowledge input is provided, FEVR is defined as 0 and RECR as 1, resulting in a KIS of 1, signifying that no restrictions are violated.

$$\text{KIS} = (1 - \text{FEVR}) \times \text{RECR} \tag{1}$$

Priority Score Another metric proposed in this work and assigned to a causal discovery graph output is the Priority Score, as detailed in Algorithm 1. It systematically organizes graphs generated by multiple causal discovery algorithms based on their potential to contain pivotal causal connections, meaning the higher the value, the more likely it contains the expected causal paths.

Graphs Comparison The generated graphs can be compared by utilizing various metrics to compare them against a ground truth graph, if available, or each other. Within *Causal-Nest* we have implemented three essential metrics for graph comparison: the Area Under the Precision-Recall Curve (AUCPR), Structural Hamming Distance (SHD) [26], and Structural Intervention Distance (SID) [18]. AUC-PRC assesses overall classification accuracy and misclassification when comparing adjacency matrices of graphs. SHD quantifies discrepancies by evaluating missing edges and directionality errors. SID examines causal relationships by computing paths between all pairs of variables to confirm consistency with causal features.

3.4 Causal Effect Estimation

Causal-Nest builds upon the DoWhy package [21] pipeline for estimating the causal effect of a variable. It is accomplished by first identifying the causal expression by assessing the graph structure and then estimating the variable effect. The proposed framework utilizes DoWhy default implementation to extract the identified estimand while disregarding unobserved confounders. Additionally, it employs DoWhy estimation methods, which can be parameterized by the user, defaulting to propensity score stratification. Therefore, the whole variable estimation process takes a causal graph and a variable as input and produces the causal estimate along with the p-value, indicating the statistical significance of the estimated causal effect.

The estimation process must be individually assessed for each variable acting as the treatment, which can lead to increased complexity. For a dataset with N independent variables and K discovered graphs, a total of $N \times K$ estimations will be carried out. In this scenario, the priority score of the discovered graphs (refer to Section 3.3) and the importance of features (refer to Section 3.2) play a crucial role in prioritizing the execution of the estimations. This prioritization is crucial as there is a hard time limit in place, which may prevent certain variables or graphs from being assessed. The estimation execution of a list of discovered graphs follows these steps:

1. Prioritize the graphs and features by sorting them in descending order based on their priority scores.
2. Create a parallelized process queue with a user-configurable maximum number of workers. By default, this maximum is set to the available CPU count in the running system.
3. Each process estimates the features sequentially, following the priority order. It has a user-configurable time limit to estimate the effect of as many variables as possible. If time runs out, partial estimations are allowed, meaning a graph can estimate a fraction of the variables.
4. After completing a process, store the estimation results independently for each model. Then, select the following graph in the queue for estimation.

3.5 Estimates Refutation

Refutation methods are built on top of the DoWhy package [21] and can easily support custom methods if they adhere to a simple interface. These methods take the dataset (and its feature mapping) and an estimation result as input and return an instance with the newly estimated effect and a p-value associated with the significance test of the newly calculated effect and the previous one. Our framework deems a refutation to have passed the refutation test if the p-value exceeds the default threshold of 0.05.

In the refutation phase, hard time limit constraints are applied due to the extensive input. Each variable serving as a treatment needs individual evaluation, and refutation must be handled for each estimation result. Multiple refutation methods are executed for each estimation result, resulting in a significant input size of $N \times K \times Z$, where N is the number of independent variables, K is the number of discovered graphs, and Z is the number of refuters. The proposed framework establishes a global timeout for the total execution time of the refuting pipeline and a model timeout for each estimation result entry, providing flexibility in handling different estimation results and refutation methods.

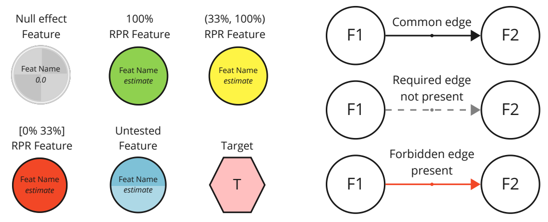
This study uses three methods of refutation: Placebo Treatment Refuter (PTR), Random Common Cause Refuter (RCCR), and Subset Removal Refuter (SRR). Each method tests different aspects of the estimated effect independently. Time constraints may limit the number of refutation tests that can be performed. The refutation execution of a list of estimation results follows these steps:

1. Prioritize the estimation results by arranging them in descending order based on their estimated effect values.
2. Create a parallelized process queue with a user-configurable maximum number of workers. By default, the maximum is set to the available CPU count.
3. Assign the process queue with independent estimation results based on the priority order. Each process tests the estimation result sequentially with the implemented refutation methods.
4. After completing the refutations for an estimated result or reaching the model-timeout threshold, store the refutation results independently for each estimation result. Then, select the next entry in the sorted list if the global timeout has not been reached.

3.6 Output Graphs Generation

The proposed framework provides a user-friendly output for quickly evaluating the generated graphs, including their estimation and refutations, as shown in Figure 2. It generates a custom DOT output graph for each graph discovered, allowing the user to inspect the causal effect estimates, the Refutation Pass Rate (RPR), and the presence of forbidden edges or the absence of required edges. Features include the feature name and the causal estimate below, while the target node is represented as a pink hexagon. Feature nodes follow a color scheme: green for variables that passed all refutation tests, red for those that passed up to 33% of the refutation tests, yellow for values in between, and split blue for variables with no refutation tests executed. A grey node is shown when the estimated effect of the node is 0, indicating that the variable has no estimated causal effect on the target.

If domain knowledge is provided as input to the framework, it may display edges in various styles in addition to the standard filled black arrow, which represents a causal relationship. A filled red arrow denotes an undesired edge, as specified in the domain input. In contrast, a gray dashed edge indicates edges listed as required in the input that were not discovered by the causal discovery algorithm. In the case of a pair of nodes with both a red and a dashed gray edge, this signifies a wrongly directed edge. *Causal-Nest* can correct these edges by modifying the graphs generated during the causal discovery process with the help of an optional flag that is turned off by default.



(a) Nodes notation for output graph (b) Edges notation for output graph

Fig. 2: Custom nodes and edges style specification for the output graph

4 Experimental Results

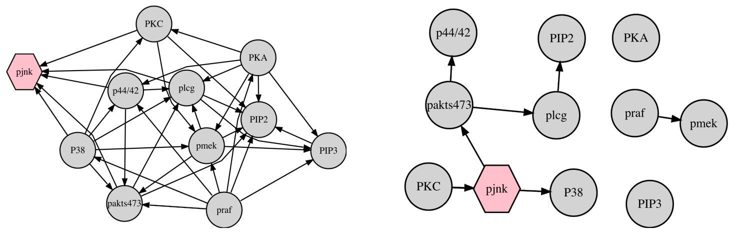
We showcase the execution of the causality pipeline in a real-world dataset that includes a generally accepted ground truth. The experiments were conducted on a 2023 MacBook Pro with an Apple M3 Pro chip and 36 GB of memory. All applied benchmarks utilized identical parameters for parallelization and time limits. Expressly, the maximum number of workers was set to 12. Additionally, a 30-minute timeout was established for each discovery model for causal discovery and estimation. A 30-minute global timeout and a 4-minute timeout per estimation result were applied for refutation.

The dataset created by Sachs *et al.* [19] is a widely used benchmark in causality due to its known ground truth graph, which consists of 19 directed edges. The dataset contains empirical quantitative data with 11 features and 7466 instances.

We treat the 'pjnk' column as the target and map all other columns as continuous features. The dataset does not contain missing data. After the input is given, the feature sorting is applied with calculated associated importance.

The ground truth graph contains two edges pointing to the target node, from PKC and PKA, as Figure 4-a shows. Even though the feature importance ranking is determined by a regressor using the data and does not consider the ground truth, our solution placed these features at the top 4 of the sorted list. Out of the 9 paths to the target node in the ground truth, 7 involve the PKC feature, and PKA has a direct edge to the target, justifying their high ranking. Furthermore, 4 out of the lowest 5 features in the ranking do not have a path to the target in the ground truth.

Out of the 12 discovery methods implemented, *Causal-Nest* identified 8 applicable: PC, GS, CCDr, IAMB, SAM, BES, GRaSP, and CGNN. Due to the dataset size and complexity, SAM and CGNN timed out, while the others yielded DAGs with various calculated priority scores. BES had the highest score at 1.04, followed by IAMB (0.79), GS (0.76), PC (0.63), GRaSP (0.56), and CCDr (0.00). Figure 3 displays the discovered graphs with the highest (a) and lowest (b) priority scores. As expected, the graph generated by BES, containing multiple paths to the target and high interconnectivity, received a higher ranking. Conversely, the final graph in the list, lacking edges pointing towards the target and disconnected, justifies its lower position in the sorting order.



(a) Discovered graph by BES (b) Discovered graph by CCDr

Fig. 3: Discovered graphs sample for Sachs dataset

Table 2 compares the metrics for the discovered graphs. Additionally, Figure 4 shows the dataset ground truth alongside the output generated by the graph with the higher estimated causal effect (generated by PC), depicting the estimated effects of the features and their associated refutations. Notably, algorithms such as CCDr and PC demonstrated competitive performance, with AUCPRC scores of 0.45 and 0.38, respectively, indicating robust predictive power in identifying causal relationships. Achieving a 100% RPR across all algorithms highlights the capability of our solution to effectively generate robust causal models, enhancing the reliability of inferred causal relationships. The distances (SHD and SID) achieved in the benchmarks demonstrate varying levels of structural accuracy across algorithms, reflecting the diverse nature of causal discovery models and indicating the utility of a diverse set of algorithms in causality problems.

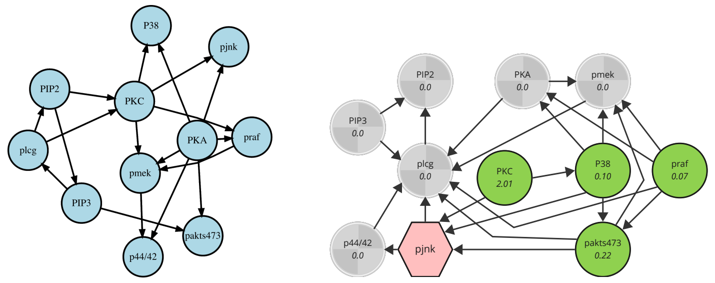
CD Model	Priority Score	AUCPR	SHD	SID	Max Causal Effect Estimate	Max Causal Effect p-value	Max Causal Effect RPR
1. GS	0.38	0.42	27	78	1.48	1.94e-05	100%
2. IAMB	0.77	0.25	35	90	1.83	2.57e-26	100%
4. CCDr	0.00	**0.45**	**17**	**72**	1.89	1.13e-18	100%
9. PC	0.63	0.38	28	83	**2.01**	1.50e-80	100%
11. GRaSP	0.18	0.09	42	86	1.87	3.98e-130	100%
12. BES	**1.04**	0.30	42	79	1.37	8.39e-114	100%

Table 2: Stats for the causality pipeline on the Sachs dataset

5 Conclusion

This work proposes a new framework that significantly advances causal inference for AI model developers, particularly those with minimal expertise in causality. By integrating multiple causal discovery algorithms and addressing crucial aspects such as time constraints and prioritization of potential causal models, *Causal-Nest* effectively simplifies the complex process of causal analysis.

Our experimental results demonstrate the capability of the framework to handle real-world problems. The performance of *Causal-Nest* on the Sachs *et*



(a) Sachs ground truth (b) *Causal-Nest* output for PC in Sachs dataset

Fig. 4: Comparison between Sachs ground truth and a generated output graph

al. [19] dataset underscores its precision in identifying known causal relationships, thereby validating its effectiveness. The framework provides user-friendly outputs, featuring custom DOT graphs and a color-coded scheme, which enhance the interpretability of causal effect estimates and refutation results, making it accessible to users with varying levels of expertise.

In conclusion, *Causal-Nest* bridges a critical gap in AI development, providing a powerful tool for uncovering and validating causal relationships. This advancement is promising to improve decision-making and predictions in sensitive and high-stakes fields such as healthcare, justice, and finance. Future work should continue to refine and expand the capabilities of the framework, ensuring its adaptability to an ever-evolving landscape of AI and causality research.

References

1. Ansel, Jason, Y.e.a.: Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In: Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2. p. 929–947. ASPLOS '24, Association for Computing Machinery (2024). `https://doi.org/10.1145/3620665.3640366`
2. Aragam, B., Gu, J., Zhou, Q.: Learning large-scale bayesian networks with the sparsebn package. Journal of Statistical Software **91**(11) (2019). `https://doi.org/10.18637/jss.v091.i11`
3. Aragam, B., Zhou, Q.: Concave penalized estimation of sparse gaussian bayesian networks. The Journal of Machine Learning Research **16**(1), 2273–2328 (2015)
4. Bühlmann, P., Peters, J., Ernest, J.: Cam: Causal additive models, high-dimensional order search and penalized regression (2014)
5. Chen, H., Harinen, T., Lee, J.Y., et al.: Causalml: Python package for causal machine learning (2020). `https://doi.org/10.48550/ARXIV.2002.11631`
6. Ensign, D., Friedler, S.A., Neville, S., et al.: Runaway feedback loops in predictive policing. In: Conference on fairness, accountability and transparency. pp. 160–171. PMLR (2018)
7. Goudet, O., Kalainathan, D., Caillou, P., et al.: Learning functional causal models with generative neural networks. Explainable and interpretable models in computer vision and machine learning pp. 39–80 (2018)

8. Hagberg, A., Swart, P.J., Schult, D.A.: Exploring network structure, dynamics, and function using networkx. Tech. rep., Los Alamos National Laboratory (LANL), Los Alamos, NM (United States) (2008)
9. Kalainathan, D., Goudet, O.: Causal discovery toolbox: Uncover causal relationships in python (2019). `https://doi.org/10.48550/ARXIV.1903.02278`
10. Kalainathan, D., Goudet, O., Guyon, I., et al.: Structural agnostic modeling: Adversarial learning of causal graphs. Journal of Machine Learning Research **23**(219), 1–62 (2022)
11. Kalisch, M., Bühlman, P.: Estimating high-dimensional directed acyclic graphs with the pc-algorithm. Journal of Machine Learning Research **8**(3) (2007)
12. Lam, W.Y., Andrews, B., Ramsey, J.: Greedy relaxations of the sparsest permutation algorithm. In: Uncertainty in Artificial Intelligence. pp. 1052–1062. PMLR (2022)
13. Margaritis, D., et al.: Learning Bayesian network model structure from data. Ph.D. thesis, School of Computer Science, Carnegie Mellon University Pittsburgh, PA, USA (2003)
14. Markus Kalisch, Martin Mächler, Diego Colombo, et al.: Causal inference using graphical models with the R package pcalg. Journal of Statistical Software **47**(11), 1–26 (2012). `https://doi.org/10.18637/jss.v047.i11`
15. Nandy, P., Hauser, A., Maathuis, M.H.: High-dimensional consistency in score-based and hybrid structure learning. The Annals of Statistics **46**(6A), 3151–3183 (2018)
16. Obermeyer, Z., Powers, B., Vogeli, C., et al.: Dissecting racial bias in an algorithm used to manage the health of populations. Science **366**(6464), 447–453 (2019)
17. Pearl, J.: Causality. Cambridge University Press (2009)
18. Peters, J., Bühlmann, P.: Structural intervention distance (sid) for evaluating causal graphs (2014)
19. Sachs, K., Perez, O., Pe'er, D., et al.: Causal protein-signaling networks derived from multiparameter single-cell data. Science **308**(5721), 523–529 (2005). `https://doi.org/10.1126/science.1105809`
20. Scutari, M.: Learning bayesian networks with the bnlearn R package. Journal of Statistical Software **35**(3), 1–22 (2010). `https://doi.org/10.18637/jss.v035.i03`
21. Sharma, A., Kiciman, E.: Dowhy: An end-to-end library for causal inference (2020). `https://doi.org/10.48550/ARXIV.2011.04216`
22. Shimizu, S., Hoyer, P.O., Hyvärinen, A., et al.: A linear non-gaussian acyclic model for causal discovery. Journal of Machine Learning Research **7**(10) (2006)
23. Spirtes, P., Glymour, C., Scheines, R.: Causation, Prediction, and Search. MIT Press (2000)
24. Syrgkanis, V., Lewis, G., Oprescu, M., et al.: Causal inference and machine learning in practice with econml and causalml: Industrial use cases at microsoft, tripadvisor, uber. In: Proceedings of the 27th ACM SIGKDD. p. 4072–4073. KDD '21, ACM (2021). `https://doi.org/10.1145/3447548.3470792`
25. Tsamardinos, I., Aliferis, C.F., Statnikov, A.R., et al.: Algorithms for large scale markov blanket discovery. In: FLAIRS. vol. 2, pp. 376–81 (2003)
26. Tsamardinos, I., Brown, L.E., Aliferis, C.F.: The max-min hill-climbing bayesian network structure learning algorithm. Machine learning **65**, 31–78 (2006)
27. Yuan, C., Malone, B.: Learning optimal bayesian networks: A shortest path perspective. Journal of Artificial Intelligence Research **48**, 23–65 (2013)
28. Zheng, Y., Huang, B., Chen, W., et al.: Causal-learn: Causal discovery in python (2023)