Causal AI for XRPL/GossipSub network configuration

Flaviene Scheidt de Cristo*, Jean-Philippe Eisenbarth*, Jorge Augusto Meira*, Radu State*
* University of Luxembourg, SnT, 29, Avenue J.F Kennedy, L-1855 Luxembourg
Email:{flaviene.scheidt, philippe.eisenbarth, jorge.meira, radu.state}@uni.lu

***Abstract*—Many peer-to-peer systems and blockchain platforms rely on underlying communication services, such as GossipSub, which typically operate with default configuration settings. A set of parameters defines these settings, and currently, there is limited understanding of how varying these parameters affects the overall service. This work proposes a methodology based on Causal AI Discovery to assess the importance of individual parameters on target indicators for the specific case of a popular p2p communication platform. We explore methods to identify factors that influence overall performance and instantiate them for the concrete case of the XRPL blockchain.**

***Index Terms*—causal analysis, causal discovery, pubsub systems, gossipsub, xrpl, network configuration**

I. Introduction

GossipSub [1] is the state-of-the-art in message propagation on blockchains. Built upon a *publisher/subscriber* (pubsub) model and using gossiping to reach distant nodes. GossipSub relies heavily on how its mesh is constructed, with the aim of scaling pubsub dissemination without excessive bandwidth or peer overload. The construction of the mesh depends on parameters that can be tuned according to the characteristics and performance of the network. Originally implemented for use in the *Interplanetary File System* (IPFS) and Ethereum, GossipSub has been shown to be a reliable and safe alternative for message dissemination on unstructured p2p networks.

In previous works [2] [3] we explored the integration of GossipSub into the XRP Ledger (XRPL) as a dissemination method to reduce message overhead and increase network scalability. The XRPL is a blockchain that uses an alternative type of consensus that largely differs from the traditional *Proof-of-Work* (PoW), using a quorum of voters instead of computational proofs. The XRP Ledger Consensus Protocol (XRP LCP) [4] behaves as a pubsub system, with each node in the network subscribing to trusted lists of peers. However, there are no deeper studies on how GossipSub behaves and how can it be tuned for integration in networks different from those for which it was originally designed.

In this work, we provide a deeper analysis of GossipSub using *causal AI discovery* methods to find causal relationships between GossipSub mesh parameters, events in the network, and performance metrics, such as message overhead, using the concrete case of the XRPL as an underlying system. We seek to answer the question of *How can causal analysis be applied as a tool to aid in the parameterization of GossipSub?*

The remainder of the paper is organized as follows: Section II presents the tools and usage of causal analysis. Then, Section III gives some background on the XRPL blockchain and its consensus protocol. In Section IV we present our methodology, and we show its results in Section V. Finally, Section VI concludes the paper.

II. Related Work

Causal analysis is a tool used in several domains, such as social sciences, biology, and medical sciences. It is not a new concept but has gained traction in recent years. To our knowledge, there is currently no work that addresses causal analysis for studying the parameterization of networks; however, the use of causal analysis to study network performance is not completely new; Hours et al. [5] present the first study to make the case for the use of graphical causal methods to analyze network performance. The work focuses on data collected from an emulated TCP network, by measuring FTP traffic, and on subsequent measurements obtained from a single FTP server connected to the Internet.

This study used the Tetrad [6] suite, concluding that constraint-based algorithms generated more informative graphs based on knowledge of the TCP domain. The particular constraint-based algorithm used to subsequently predict network performance was the PC algorithm [7]. However, at the time of the study, Tetrad assumed either normality or linearity on the data distribution. In our work, we considered four classes of discovery algorithms, including gradient-based ones that were not yet available at the time of the Hours et al. study.

III. Background

A. GossipSub

GossipSub [1] has originally been proposed as a solution for the dissemination of blocks and transactions on FileCoin and Ethereum 2.0. The tool is distributed as an extensible component inside libp2p [8]. Pubsub systems, as the name suggests, have two primary entities: publishers and subscribers. Messages are published on topics, and entities can subscribe to these topics to receive messages.

We can abstract GossipSub as being formed by two overlays, the first being a *full-message overlay* and the second a *gossip layer* where metadata is disseminated to serve as a set reconciliation phase to achieve distant or poorly connected nodes.

The construction of the GossipSub mesh is given by a set of parameters [8] that control both the full-message and the gossip overlay. The primary parameters are the so-called *d-values*. From those, only one – *Dlazy* – is not linked to the full-message overlay. In this work, we focus on the full-message overlay, choosing the message overhead metric as a target for causal analysis.

D is the target value of how many peers a node should connect to *per subscribed topic*. This value can be relaxed based on the network conditions according to a minimum and a maximum, *Dlo* and *Dhi*. Forming a new connection is called a *graft* and a deliberate disconnection is called a *prune*. Both *Dlo* and *Dhi* can trigger these two events. When the number of peers to which a given node is connected increases above *Dhi* for a given topic, a series of prune events are triggered. The node not only prunes enough nodes to satisfy *Dhi* but also keeps only a *Dout* number of outbound connections per topic. The node then grafts connections to keep the number of peers above *Dlo*, choosing *Dscore* high-scored peers and connecting to the rest randomly.

B. The XRP Ledger Consensus Protocol

The XRPL was one of the first blockchains to be released, presenting a novelty type of consensus that differs from the traditional PoW [4]. The XRP LCP uses quorum-based voting over several rounds to validate new ledger versions. Each network validator has a list of trusted nodes called a *Unique Nodes List* (UNL). Any node can claim to be a validator, able to propose new ledger versions; however, not all nodes are trusted by the network.

The nodes take into account the position of the peers present in their UNLs to adjust their ledger proposals and validate new ledger versions. In this way, we can call the XRPL an inherently pubsub system, abstracting UNLs as topics in which validators declare their trust by subscribing. In previous work, we explored the pubsub [2] characteristics of the XRP LCP, suggesting its integration with GossipSub to broadcast proposals and validations with the goal of reducing message overhead and increasing network scalability.

IV. Methodology

A. Steps of the Causal Analysis

To better understand the steps used in the methodology of this work, we use the *Ladder of Causation*, an abstract concept introduced by Judea Pearl [9] in which a ladder represents levels of relationships between the dimensions of the scenario being analyzed. The first step of the ladder is called *Association*, and is where we *observe* the behavior of the system from the perspective of how one variable changes another. To find relationships, we used the two previous studies conducted on GossipSub over the XRPL [2] [3]. The first focuses on employing GossipSub to mitigate message overhead on the XRPL, using different types of topic arrangements on the GossipSub level. The second is a multidimensional analysis of the parameters used to build the GossipSub mesh, focusing on the correlation between the dimensions formed by the parameters and the measurement of events and metrics of performance. These works helped us hypothesize relationships between the parameters and the resulting behavior of the network.

The second step of the ladder is the *intervention*, and it is where we make changes to the parameterization of the system to understand how a variable can affect another. In this phase, we conducted randomized experiments using a testnet of 24 nodes running instances of *Flexi-pipe* [2], an agnostic tool that allows us to plug different dissemination techniques into the XRPL validator. We use GossipSub to propagate proposals, randomly selecting 44 sets of parameters[1] using three structures of topics.

Structure **1** uses one global topic to disseminate validations, with all nodes publishing on this topic and all nodes also subscribed to this topic. In the context of the XRPL, this structure is similar to how the *Mainnet* is currently implemented, with a unique UNL to guarantee safety and liveness [4], also similar to how Ethereum disseminates blocks [1]. Structure **2** has 24 topics of size 1, with each node subscribed to 16 nodes on average, with a maximum of 18 and a minimum of 15 subscriptions, abstracting each node as a topic. The last structure, **3**, uses eight predefined topics, each node subscribed to 1 topic, the smallest topic having 16 nodes and the largest 18 nodes. The last two structures have been previously studied [2] [3] as ways to improve the message overhead through enhancing the pubsub characteristics of GossipSub.

We still miss the third step of the ladder: *counterfactual*. This phase requires us to push the system into behaving in ways that have not previously been observed. This is done by *hypothetical* or *simulated* interventions [10]. Those interventions are different from the ones described previously, in a way that they are not interventions done in the system, but in model-scenarios created in the two previous steps. In this work, we focus on the first two steps, leaving the counterfactual analysis as the next stage, considering that a poorly adjusted model negatively impacts the simulation of interventions.

B. Generating Causal Graphs from Observational Data

We used observational data and domain knowledge to generate a causal graph for GossipSub, considering parameters related to mesh construction and targeting one measurement related to scalability and performance: *messageOverhead*. We used the findings of a previous study [3] on the correlation between the GossipSub mesh parameters and the performance of the system to identify the parameters that were correlated with the target measurement. We then created the causal graph, showed in Figure 1, to express only the relationships that impact the target, namely most of the parameters related to the *full-message* overlay. Considering that most of the

[1]The list of parameters set is available at https://github.com/FlavScheidt/causalGossipSub/blob/main/Datasets/parameters.csv

parameters that impact *messageOverhead* are defined per topic, we added two variables related to the number of topics and topic size, respectively *topics* and *topicSize*. In this scenario, the parameters are *D*, *Dlo*, *Dhi*, *Dout*, *Topics* and *TopicSize*.

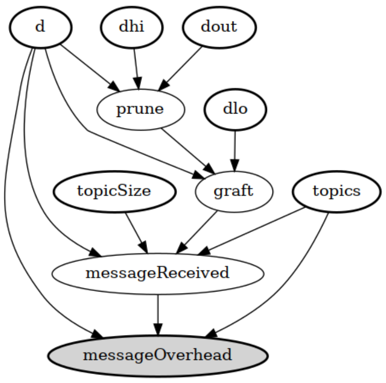

Fig. 1. Causal graph generated with observational data

D determines the optimal number of connections per topic in the full-message overlay and is more likely to impact the number of *grafts* and *prunes*. It can also lead to changes in the number of *messagesReceived*; regarding *messageOverhead*, *D* affects the number of replicas a node gets, forwarded by their direct peers. *Dhi* influences the *prune* of connections, as it sets the limit for the number of connections per topic. During a *prune*, only *Dout* connections from high-scoring peers are retained, and new connections are then grafted to satisfy *Dlo*. Thus, *prune* events typically trigger *graft* events, which are also related to *Dlo*. The number of *topics* also impacts *messagesReceived* and *messageOverhead*, as each *d-value* is specified per topic. The *topicSize* influences the total number of *messagesReceived*, directly affecting the *messageOverhead*.

C. Generating Causal Graphs from Interventional Data

Moving to the second step of the ladder, we performed interventions in the system to analyze the impact of changes in parameters on the target measurement. The interventions were carried out using a cluster with 24 nodes that formed a private XRPL testnet using three different topic structures, as explained in Section IV-A. For each set in each structure, we performed three tests of 30 minutes, ignoring faulty executions. We consider faulty the executions in which the number of events recorded is below a z-score of $0, 15 * standard\, deviation$. Data were collected from the GossipSub event tracer and events were grouped by 5-second increments.

We used the gCastle [11] package to perform causal discovery in the dataset obtained[2]. gCastle is a Python toolbox for learning causal structures, implementing 19 algorithms from different categories, and also providing evaluation metrics for the generated graphs. From the algorithms provided, we chose 3: PC [7] [6], LiNGAM [12], and NOTEARS [13]. We chose these algorithms to test the different categories of causal discovery: *constraint-based* (PC), *function-based* (LiNGAM), and *gradient-based* (NOTEARS).

PC has the advantage of being the only algorithm implemented in gCastle that accepts the input of prior knowledge in the causal discovery in the form of required and forbidden edges. We applied three variants of PC: classical [14], parallel [15] and stable [16], inputting prior knowledge in the form of forbidden edges, preventing the algorithm from trying to find relationships between the parameters and in a reverse causal way, meaning that we prevent the creation of edges from metrics resulting in parameters.

LinGaM and NOTEARS do not accept prior knowledge; both found relationships between parameters that are true from a correlational point of view, considering that some *d-values* are constrained by the value of *D*. LinGaM also found spurious relationships between *topicSize* and *topics*. We pruned these edges for subsequent evaluations as they are not truly causal relationships. The algorithms did not find any reverse causal connection; that is, in NOTEARS and LinGaM, the parameters always cause the metrics and not vice versa. We used three variations of NOTEARS: pure NOTEARS, LowRank [17] and GOLEM [18] and two of LinGaM: ICA [19] and Direct [20].

D. Generating Graphical Causal Models

The previous sections described how to obtain an SCM in the form of a causal DAG, we now go one step further and assign causal mechanisms for each one of the variables in the model to generate a Graphical Causal Model (GCM). This assignment is made by assuming a distribution for the variables based on a correlation matrix. We used a Python library for causal analysis called DoWhy [21] to automatically assign causal mechanisms to our models. DoWhy assigned discrete distribution to all non-root nodes, and additive noise model to all variables in the model.

V. Evaluation

A. Refuting Causal Graphs

Table I shows the first evaluation metrics, obtained by gCastle for the graphs generated by each method. However, these evaluation metrics are generated on the basis of the underlying truth, which is represented by the causal graph generated during the observation phase (Figure 1). The metrics in Table I then give us the distance between the model generated with observational data using domain knowledge and the model generated by interventional data using discovery algorithms.

The metrics in Table I are only sufficient to evaluate the effectiveness of the discovery algorithms if we assume that the graph generated using observational data is true in representing the causal relationships of the system. At this point, we do not know whether the observational graph is consistent with the interventional data. This question cannot yet be answered directly. However, we have tools to refute

[2]Code and graphs available at https://github.com/FlavScheidt/causalGossipSub/tree/main/1_Discovery

TABLE I
GCastle Evaluation of Discovery Algorithms

	Precision	Recall	F1
PC Classical	0.42	0.69	0.52
PC Stable	0.42	0.69	0.52
PC Parallel	0.42	0.69	0.52
Direct LinGaM	0.31	0.38	0.34
ICA LinGaM	0.43	0.53	0.48
NOTEARS	0.41	0.53	0.46
NOTEARS LowRank	0.21	0.38	0.27
NOTEARS GOLEM	0.2	0.23	0.21

a certain causal graph if it does not satisfy some conditional independence statements on its nodes called *Local Markov Conditions* (LMCs) [22]. We again use DoWhy to refute the generated graphs. DoWhy provides tools to abstract the causal reasoning process, making it more accessible to non-experts. One of the key features of the package is the ability to evaluate models, providing an overview of different metrics that provide information on the performance of the causal model [23].

DoWhy can perform independence tests on separate sets of variables, but given the size and amount of graphs we wish to evaluate, we chose to use another tool from the package: graph falsification[3] [24]. The falsification tool gives the summary of the number of LMC violations, expressed by the p-value. This summary represents the comparison between the number of LMC violations against randomly generated graphs.

We do not refute the graph; instead, we use the p-values as a baseline for the evaluation of the causal discovery methods in Section V; the p-value for LMC being 0.1. This does not mean that the graph is a perfect representation of the system but implies that the graph is better than 90% of the randomly generated graphs. To evaluate the performance and fitness of the causal models generated by observational and interventional data, we first generated the falsification summaries for the SCMs, shown in Table II.

TABLE II
Falsification Summary of Discovery Algorithms

	LMC	LMC Violations	LMC Violations Rate
Observational	0.10	23/52	0.44
PC Classical	0.65	21/45	0.47
PC Stable	0.20	18/45	0.40
PC Parallel	0.35	22/45	0.49
D LinGaM	0.05	31/68	0.46
ICA LinGaM	0	29/64	0.45
NOTEARS	0.10	29/66	0.44
NT LowRank	0.05	37/64	0.42
NT GOLEM	0.05	42/78	0.54

DoWhy uses a threshold of 0.05 for the LMC p-value to refute graphs. By this estimation, only Direct LinGaM, ICA LinGaM, NOTEARS LowRank, and NOTEARS GOLEM would not be rejected. However, this threshold can be arbitrary in a real-world scenario, and so we do not assume any threshold to refute graphs, using the LMC p-value as a comparison metric, instead. We can see that ICA LinGaM has the ideal LMC p-value but still shows LMC violations, with PC Stable having the lowest rate of violations. We consider these violations to be present because of latent variables, that is, variables that cannot be observed but have a greater impact than some observed variable [5].

[3]Code and evaluation reports available at https://github.com/FlavScheidt/causalGossipSub/tree/main/2_ModelEvaluation

B. Performance Evaluation

DoWhy can also evaluate GCMs to verify how well the models perform, if the assumption of the addictive noise model is correct, and how well the GCM captures the joint distribution of the observed data [23]. We chose two metrics in our evaluation, shown in Table III. First, we look at the overall average KL-divergence (KL-Div in the table) between the generated and the observed distributions, the lower the KL-divergence, the better the model fits the observed data distribution. The second metric is obtained by non-root node, evaluating the accuracy of the causal mechanisms expressed in the model, and is measured by the normalized *Continuous Ranked Probability Score* (CRPS); the closer this value is to zero, the better the precision of the causal mechanism for that given node.

It is important to note that the results for LinGaM and NOTEARS – including all variations – had bigger KL-divergences before the pruning of the edges that represented relationships between the parameters. The values were around 7 before pruning and then decreased to around 3.7 to 2.5 afterward. We suppressed this analysis from this work for the sake of brevity, but it shows once again the importance of domain knowledge in the modeling of causal structures.

From the metrics represented in Table III, we can see that NOTEARS LowRank and GOLEM have the smallest KL-divergence. However, both models ended up excluding *graft* and *prune* events, which may indicate that these events have little to no effect on *messageOverhead*. GOLEM has a high CRPS for *messageReceived*, making LowRank a better candidate. Nonetheless, we need to consider that both algorithms have low F1 scores (Table I) when considering the domain knowledge model. So we turn our attention to the LinGaM variations and pure NOTEARS.

From Table II, ICA LinGaM showed to better fit the data, and also showed decent KL-divergence while having good or very good CRPS for the non-root nodes, however showing a worst CRPS for *messageOverhead* compared Direct LinGaM and NOTEARS. Between those last two, NOTEARS showed a lower CRPS for the target measurement, keeping a KL-Divergence close to ICA LinGaM and an acceptable F1 score. Therefore, using the evaluation tools and domain knowledge, NOTEARS generated a suitable causal model for the proposed scenario.

VI. Discussion & Perspectives

In this work, we presented an analysis of causal AI discovery methods to find graphical causal models that represent the relationship between the parameters and the performance

TABLE III
DoWhy Evaluation

		CRPS			
	KL-Div	**graft**	**prune**	**messageReceived**	**messageOverhead**
Observational	4.88	0.14	0.88	0.35	0.18
PC Classical	5.06	0.15	0.06	0.11	0.35
PC Stable	5.07	0.15	0.06	0.11	0.35
PC Parallel	5.07	0.15	0.06	0.11	0.35
Direct LinGaM	3.78	0.25	-	0.35	0.13
ICA LinGaM	3.61	0.25	-	0.37	0.18
NOTEARS	3.62	0.25	-	0.37	0.13
NOTEARS LowRank	2.50	-	-	0.33	0.13
NOTEARS GOLEM	2.51	-	-	0.35	0.14

of a pubsub-based network. We aim to address how causal analysis can assist in parameterizing networks, specifically GossipSub, to disseminate ledger proposals over the XRP Ledger.

The first two steps of causal analysis are the most challenging ones; being where we discover causal relations by looking at observational and/or interventional data. This work focused on this discovery by applying three categories of algorithms – *constraint-based*, *function-based*, and *gradient-based* – in contrast to a pure domain knowledge-based approach over observational data.

Gradient-based and function-based models had better accuracy and higher simplicity. This is given not only by the accuracy metrics acquired but also by the proximity between the graphs generated by NOTEARS and the model generated using observational data and domain knowledge.

This work sets a foundation upon which we can better understand the mechanisms of cause and effect that the GossipSub mesh parameters have over network performance. Future work can be focused on causal inference and climb to the third step of the causation ladder, the counterfactual. A good model can give us the tools necessary to better parameterize the network for maximum performance without the need for extensive experimentation.

Acknowledgment

We thankfully acknowledge the support from the RIPPLE University Blockchain Research Initiative (UBRI) for our research.

References

[1] D. Vyzovitis, Y. Napora, D. McCormick, D. Dias, and Y. Psaras, “Gossipsub: Attack-resilient message propagation in the filecoin and ETH2.0 networks,” 2020. [Online]. Available: https://arxiv.org/abs/2007.02754

[2] F. Scheidt de Cristo, W. M. Shbair, L. Trestioreanu, and R. State, “Pub/sub dissemination on the xrp ledger,” in *2023 IEEE Latin-American Conference on Communications (LATINCOM)*. IEEE, 2023, best Paper Award.

[3] F. Scheidt de Cristo, J.-P. Eisenbarth, J. A. Meira, and R. State, “A 9-dimensional analysis of gossipsub over the xrp ledger consensus protocol,” in *NOMS 2024 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2024.

[4] B. Chase and E. MacBrough, “Analysis of the xrp ledger consensus protocol,” 2018. [Online]. Available: https://arxiv.org/abs/1802.07242

[5] H. Hours, E. Biersack, and P. Loiseau, “A causal approach to the study of tcp performance,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 7, no. 2, pp. 1–25, 2015.

[6] P. Spirtes, C. N. Glymour, and R. Scheines, *Causation, prediction, and search*. MIT press, 2000.

[7] P. Spirtes and C. Glymour, “An algorithm for fast recovery of sparse causal graphs,” *Social science computer review*, vol. 9, no. 1, pp. 62–72, 1991.

[8] “gossipsub v1.0: An extensible baseline pubsub protocol,” accessed: 2024-03-20. [Online]. Available: https://github.com/libp2p/specs/blob/master/pubsub/gossipsub/gossipsub-v1.0.md

[9] J. Pearl and D. Mackenzie, *The book of why: the new science of cause and effect*. Basic books, 2018.

[10] A. Molak, *Causal Inference and Discovery in Python: Unlock the secrets of modern causal machine learning with DoWhy, EconML, PyTorch and more*. Packt Publishing Ltd, 2023.

[11] K. Zhang, S. Zhu, M. Kalander, I. Ng, J. Ye, Z. Chen, and L. Pan, “gcastle: A python toolbox for causal discovery,” 2021.

[12] P. O. Hoyer and A. Hyttinen, “Bayesian discovery of linear acyclic causal models,” *ArXiv*, vol. abs/1205.2641, 2009. [Online]. Available: https://api.semanticscholar.org/CorpusID:11717717

[13] X. Zheng, B. Aragam, P. K. Ravikumar, and E. P. Xing, “Dags with no tears: Continuous optimization for structure learning,” *Advances in neural information processing systems*, vol. 31, 2018.

[14] M. Kalisch and P. Bühlman, “Estimating high-dimensional directed acyclic graphs with the pc-algorithm.” *Journal of Machine Learning Research*, vol. 8, no. 3, 2007.

[15] T. D. Le, T. Hoang, J. Li, L. Liu, H. Liu, and S. Hu, “A fast pc algorithm for high dimensional causal discovery with multi-core pcs,” *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 16, no. 5, pp. 1483–1495, 2016.

[16] D. Colombo, M. H. Maathuis *et al.*, “Order-independent constraint-based causal structure learning.” *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3741–3782, 2014.

[17] Z. Fang, S. Zhu, J. Zhang, Y. Liu, Z. Chen, and Y. He, “On low-rank directed acyclic graphs and causal structure learning,” *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[18] I. Ng, A. Ghassami, and K. Zhang, “On the role of sparsity and dag constraints for learning linear dags,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 943–17 954, 2020.

[19] S. Shimizu, P. O. Hoyer, A. Hyvärinen, A. Kerminen, and M. Jordan, “A linear non-gaussian acyclic model for causal discovery.” *Journal of Machine Learning Research*, vol. 7, no. 10, 2006.

[20] S. Shimizu, T. Inazumi, Y. Sogawa, A. Hyvarinen, Y. Kawahara, T. Washio, P. O. Hoyer, K. Bollen, and P. Hoyer, “Directlingam: A direct method for learning a linear non-gaussian structural equation model,” *Journal of Machine Learning Research-JMLR*, vol. 12, no. Apr, pp. 1225–1248, 2011.

[21] A. Sharma and E. Kiciman, “Dowhy: An end-to-end library for causal inference,” *arXiv preprint arXiv:2011.04216*, 2020.

[22] D. Janzing and B. Schölkopf, “Causal inference using the algorithmic markov condition,” *IEEE Transactions on Information Theory*, vol. 56, no. 10, pp. 5168–5194, 2010.

[23] “Dowhy: Evaluate a gcm,” accessed: 2024-03-20. [Online]. Available: https://www.pywhy.org/dowhy/v0.11.1/user_guide/modeling_gcm/model_evaluation.html

[24] E. Eulig, A. A. Mastakouri, P. Blöbaum, M. Hardt, and D. Janzing, “Toward falsifying causal graphs using a permutation-based test,” *arXiv preprint arXiv:2305.09565*, 2023.