Redefining the Shortest Path Problem Formulation of the Linear Non-Gaussian Acyclic Model: Pairwise Likelihood Ratios, Prior Knowledge, and Path Enumeration

Hans Jarett J. Ong

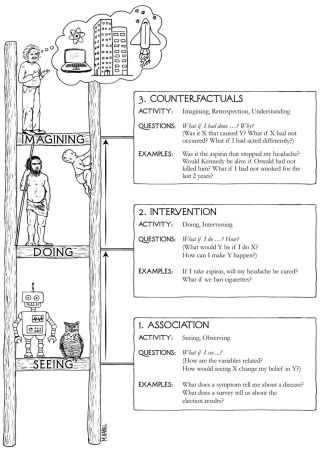
Mathematical Informatics Laboratory
Division of Information Science
Nara Institute of Science and Technology

June 18, 2024

Outline

1. Towards LiNGAM as a Shortest Path Problem (LiNGAM-SPP)
   - Graphical Causality
   - Linear Non-Gaussian Acyclic Models (LiNGAM)
   - LiNGAM-SPP
2. **Improving LiNGAM-SPP** using Pairwise Likelihood Ratios
   - Tests on Simulated Data
   - Tests on Real-world Datasets
3. **Extension 1: Incorporating Known Relative Ordering** via a node-skipping strategy
4. **Extension 2: Predicting Causal Graph Properties**
   - Utilizing the Path Distribution of all Causal Orderings
   - Predicting Confounder Presence
   - Predicting Causal Graph Sparseness
   - Predicting LiNGAM/ LiNGAM-SPP Performance

Graphical Causality



Pearl & Mackenzie (2018)

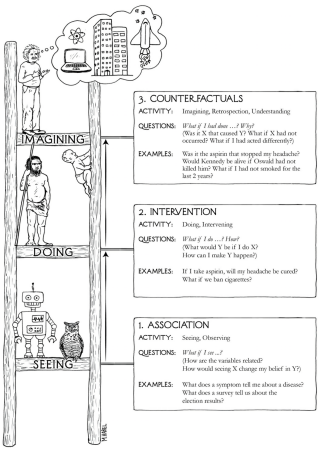
- Level 1: Association
- Level 2: Interventions
  - Do-calculus, e.g. $P(Y|do(X = x))$
  - Given some causal graph $\mathcal{G}$:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|\mathrm{PA}_i^{\mathcal{G}})$$

  - e.g. Average Treatment Effects
- Level 3: Counterfactuals
  - e.g. $PNS = P(Y_{X=0} = 0, Y_{X=1} = 1)$
  - Uses Structural Causal Models (SCMs). For some causal graph $\mathcal{G}$

$$X_i := f_i(\mathrm{PA}_i^{\mathcal{G}}, U_i).$$

Graphical Causality



Pearl & Mackenzie (2018)

- Causal Inference
  - Given a *causal graph* and data, estimate causal quantities.
- **Causal Discovery**
  - Given data (observational or interventional), learn the underlying *causal graph*.
  - relies of assumptions about the data generating process
  - e.g. causal faithfulness, structural form like Additive Noise Models and **LiNGAM**

Linear Non-Gaussian Acyclic Models (LiNGAM)

- SCM restricted to linear models and non-Gaussian noise:

$$\mathbf{x} = \mathbf{B}\mathbf{x} + \mathbf{e}$$

- Algorithms/ Methods:
  - ICA-LiNGAM (Shimizu et al. (2006))
  - **DirectLiNGAM** (Shimizu et al. (2011))

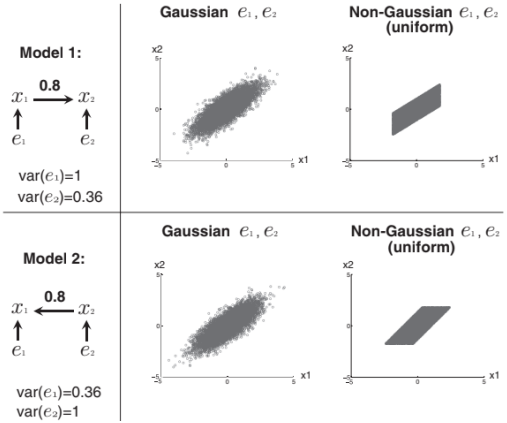


Figure from Shimizu (2014).

Linear Non-Gaussian Acyclic Models (LiNGAM)

- DirectLiNGAM steps:
  1. Get the **causal order**.
  2. Remove edges with adaptive lasso.
- e.g. The the causal ordering of $\{X, Y, Z\}$, we get a fully connected DAG

$$\begin{cases} X := \epsilon_1 \\ Y := aX + \epsilon_2 \\ Z := bX + cY + \epsilon_3 \end{cases}$$

- We can think of DirectLiNGAM as a *greedy algorithm* that determines the causal order in a fixed number of steps.

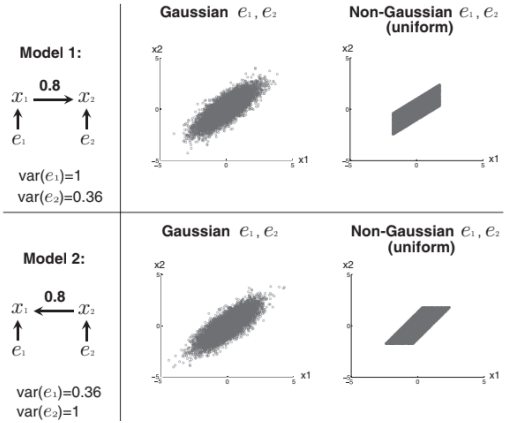


Figure from Shimizu (2014).

Linear Non-Gaussian Acyclic Models (LiNGAM)

DirectLiNGAM

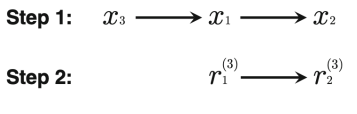


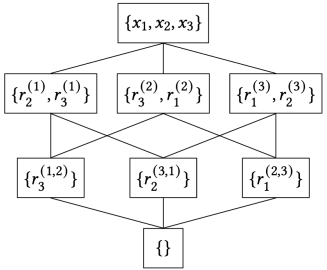
Figure 7: An illustration of DirectLiNGAM: $r_2^{(3,1)}$ denotes the residual when $r_2^{(3)}$ is regressed on $r_1^{(3)}$.

Figure from Shimizu (2014)

$$r_i^{(j)} = x_i - \frac{cov(x_i, x_j)}{var(x_j)} x_j$$

$$\begin{cases} x_3 := \epsilon_3 \\ x_1 := \alpha x_3 + \epsilon_1 \implies r_1^{(3)} = x1 - \alpha x_3 \\ x_2 := \beta x_1 + \gamma x_3 + \epsilon_2 \implies r_2^{(3)} = x_2 - \gamma x_3 \end{cases}$$

LiNGAM as a Shortest Path Problem (LiNGAM-SPP)



Adapted from Suzuki & Yang (2022).

- Each of the $n!$ paths corresponds to a causal ordering
- This takes advantage of the property:

$$I(X, Y, Z) = I(\{X, Y\}, Z) + I(X, Y)$$

- The path $\{x_1, x_2, x_3\} \rightarrow \{r_2^{(1)}, r_3^{(1)}\} \rightarrow \{r_3^{(1,2)}\} \rightarrow \{\}$ corresponds to the causal ordering $\{X_1, X_2, X_3\}$ and is computed by adding the edges

$$I(\epsilon_1, \epsilon_2, \epsilon_3) = I(x_1, \{r_2^{(1)}, r_3^{(1)}\}) + I(r_2^{(1)}, r_3^{(1,2)}) + 0$$

- Dijkstra's algorithm with lazy evaluation was used.

Outline

1. ~~Towards LiNGAM as a Shortest Path Problem (LiNGAM-SPP)~~
   - ~~Graphical Causality~~
   - ~~Linear Non-Gaussian Acyclic Models (LiNGAM)~~
   - ~~LiNGAM-SPP~~
2. **Improving LiNGAM-SPP** using Pairwise Likelihood Ratios
   - Tests on Simulated Data
   - Tests on Real-world Datasets
3. **Extension 1: Incorporating Known Relative Ordering** via a node-skipping strategy
4. **Extension 2: Predicting Causal Graph Properties**
   - Utilizing the Path Distribution of all Causal Orderings
   - Predicting Confounder Presence
   - Predicting Causal Graph Sparseness
   - Predicting LiNGAM/ LiNGAM-SPP Performance

Improving LiNGAM-SPP

- The initial LiNGAM-SPP paper used a kNN-based mutual information estimator. (Suzuki & Yang (2022))
- A subsequent study used Copula Entropy. (Suzuki & Yang (2024))
- Both methods relied on tuning the k parameter.
- We propose using **Pairwise Likelihood Ratios (PLR)** instead. (Hyvärinen & Smith (2013)). PLR doesn't require parameter tuning.
- We adapted the PLR implementation from the `lingam` package. (Ikeuchi et al. (2023))

Tests on Simulated Data

We compare the following methods:

- PLR LiNGAM-SPP (our method)
- PLR DirectLiNGAM (default implementation in `lingam` package)
- Copent LiNGAM-SPP* (from Suzuki & Yang (2024))
- kNN LiNGAM-SPP* (from Suzuki & Yang (2022))

Using the ordering error as the metric:

$$E_o = \frac{2r}{p(p-1)},$$

where p represents the number of features and r is the count of pairs in the wrong order.

*for the sake of comparison, we take the best performing k among $0.05N$, $0.1N$, and $\sqrt{N}$ where N is the sample size of the simulation.

Steps to Generate Simulations

This was patterned after Shimizu et al. (2011), modified to include confounders.

1. We initialized the matrix $\mathbf{B}$ by creating a $p \times p$ lower triangular matrix with elements randomly drawn from a uniform distribution in the interval $[-1.5, -0.5] \cup [0.5, 1.5]$, following Silva et al. (2006).
2. To introduce sparsity into the graph, we applied an element-wise product with a Bernoulli distribution matrix, following the approach of Kalisch & Bühlmann (2007). The success probability parameter of this Bernoulli distribution determines the sparseness of the graph.
3. For modeling non-Gaussian noise and confounders, we used the set of 18 non-Gaussian distributions used in Bach & Jordan (2003). Each noise term and confounder were randomly sampled from this set. Moreover, we scaled these distributions to ensure their variances fell within the interval $[1, 3]$, as in Silva et al. (2006).
4. Construct $\mathbf{\Lambda}$ sequentially. Me make sure that each confounder must confound at least 2 variables and that no 2 confounders can confound the same set of variables.

Simulation Results: Average Ordering Error

		Without Confounders				With Confounders			
	Sample Size (N)	100	250	500	1000	100	250	500	1000
p	Method								
5	PLR LiNGAM-SPP	0.29	0.28	0.23	0.21	0.46	0.45	0.42	0.39
	PLR DirectLiNGAM	0.29	0.28	0.23	0.22	0.46	0.45	0.42	0.40
	Copent LiNGAM-SPP	0.40	0.41	0.38	0.39	0.43	0.40	0.43	0.42
	*k*NN LiNGAM-SPP	**0.29**	**0.23**	**0.22**	**0.18**	**0.40**	**0.37**	**0.35**	**0.33**
10	PLR LiNGAM-SPP	0.33	**0.24**	**0.21**	**0.18**	0.44	**0.40**	**0.37**	**0.34**
	PLR DirectLiNGAM	**0.32**	0.25	0.22	0.19	**0.44**	0.41	0.37	0.36
	Copent LiNGAM-SPP	0.49	0.48	0.50	0.49	0.46	0.46	0.45	0.46
	*k*NN LiNGAM-SPP	0.48	0.40	0.35	0.30	0.49	0.43	0.39	0.35
15	PLR LiNGAM-SPP	0.33	**0.24**	**0.19**	**0.16**	**0.41**	**0.36**	0.35	**0.29**
	PLR DirectLiNGAM	**0.32**	0.25	0.21	0.18	0.42	0.37	**0.33**	0.32
	Copent LiNGAM-SPP	0.51	0.50	0.51	0.52	0.48	0.48	0.49	0.49
	*k*NN LiNGAM-SPP	0.56	0.51	0.45	0.38	0.54	0.49	0.46	0.40

Simulation Results: Average Runtime

		Without Confounders				With Confounders			
	Sample Size (N)	100	250	500	1000	100	250	500	1000
p	Method								
5	PLR LiNGAM-SPP	0.44s	0.44s	0.45s	0.46s	0.43s	0.45s	0.44s	0.46s
	PLR DirectLiNGAM	0.04s	0.04s	0.04s	0.05s	0.04s	0.04s	0.04s	0.05s
	Copent LiNGAM-SPP	0.06s	0.11s	0.3s	1.1s	0.07s	0.13s	0.35s	1.3s
	kNN LiNGAM-SPP	0.27s	0.39s	0.72s	1.8s	0.26s	0.37s	0.65s	1.7s
10	PLR LiNGAM-SPP	6.8s	6.9s	6.9s	6.1s	6.4s	7.2s	7s	7.6s
	PLR DirectLiNGAM	0.26s	0.27s	0.3s	0.36s	0.26s	0.27s	0.31s	0.36s
	Copent LiNGAM-SPP	0.4s	0.6s	1.3s	4.6s	0.48s	0.7s	1.6s	5.5s
	kNN LiNGAM-SPP	32s	45s	1.4m	3.5m	32s	44s	1.3m	3.4m
15	PLR LiNGAM-SPP	2.2m	1.6m	1.2m	1.5m	1.4m	3.6m	2.8m	3.2m
	PLR DirectLiNGAM	0.83s	0.9s	0.98s	1.2s	0.84s	0.92s	1s	1.2s
	Copent LiNGAM-SPP	1.3s	1.8s	3.4s	11s	1.6s	2.1s	4.3s	14s
	kNN LiNGAM-SPP	37m	48m	84m	206m	36m	49m	85m	205m

Tests on Real-World Datasets

- We use the following datasets from the CMU CLeaR Group:
  - sachs (Sachs et al. (2005))
  - yacht-hydrodynamics (Gerritsma et al. (2013))
  - abalone (Nash et al. (1995))
  - airfoil-self-noise (Brooks et al. (2014))
  - wine-quality-red (Cortez et al. (2009))
  - wine-quality-white (Cortez et al. (2009))
- LiNGAM-SPP is only concerned with causal ordering, we used the same sparse regression technique as DirectLiNGAM for edge removal.
- Ground Truth contains information on required and forbidden edges.

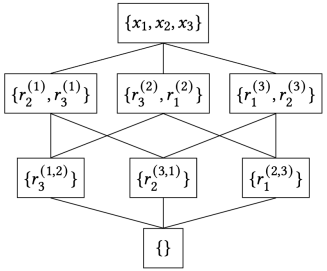
Results on CMU-CleaR Data

		Known Edges		DirectLiNGAM		LiNGAM-SPP	
Dataset	p	Req.	Forb.	Req.	Forb.	Req.	Forb.
sachs	11	20	0	10	0	10	0
yacht-hydrodynamics	7	1	15	0	2	0	2
abalone	8	0	17	0	4	0	4
airfoil-self-noise	6	0	11	0	6	0	6
wine-quality-red	12	2	11	1	7	2	2
wine-quality-white	12	2	11	2	4	2	1

Outline

1. ~~Towards LiNGAM as a Shortest Path Problem (LiNGAM-SPP)~~
   - ~~Graphical Causality~~
   - ~~Linear Non-Gaussian Acyclic Models (LiNGAM)~~
   - ~~LiNGAM-SPP~~
2. ~~Improving LiNGAM-SPP using Pairwise Likelihood Ratios~~
   - ~~Tests on Simulated Data~~
   - ~~Tests on Real-world Datasets~~
3. **Extension 1: Incorporating Known Relative Ordering** via a node-skipping strategy
4. **Extension 2: Predicting Causal Graph Properties**
   - Utilizing the Path Distribution of all Causal Orderings
   - Predicting Confounder Presence
   - Predicting Causal Graph Sparseness
   - Predicting LiNGAM/ LiNGAM-SPP Performance

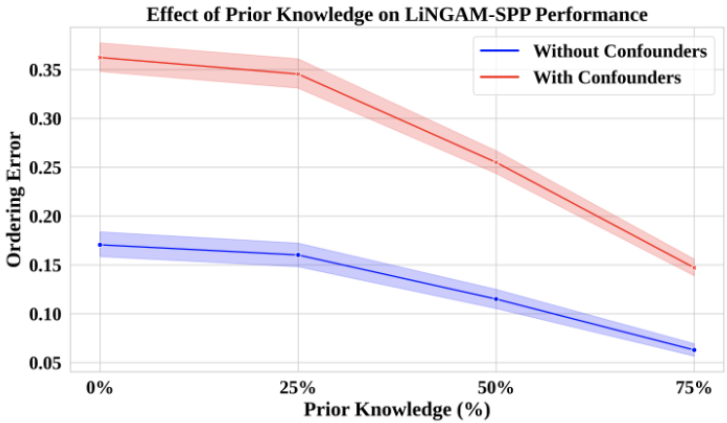
Incorporating Known Relative Ordering



Adapted from Suzuki & Yang (2022).

- In DirectLiNGAM, prior knowledge is given as an adjacency matrix where we specify edges as **required, forbidden, or unknown**.
- The path formulation allows for **relative ordering**, which is more flexible.
- Feature indices not in the subscripts are "earlier" in the ordering.
- e.g. if x_1 preceeds x_2, then the nodes with subscript 1 but not 2 ($\{r_3^{(2)}, r_1^{(2)}\}$ and $\{r_1^{(2,3)}\}$) should be skipped.

Results: Average Error vs Prior Knowledge

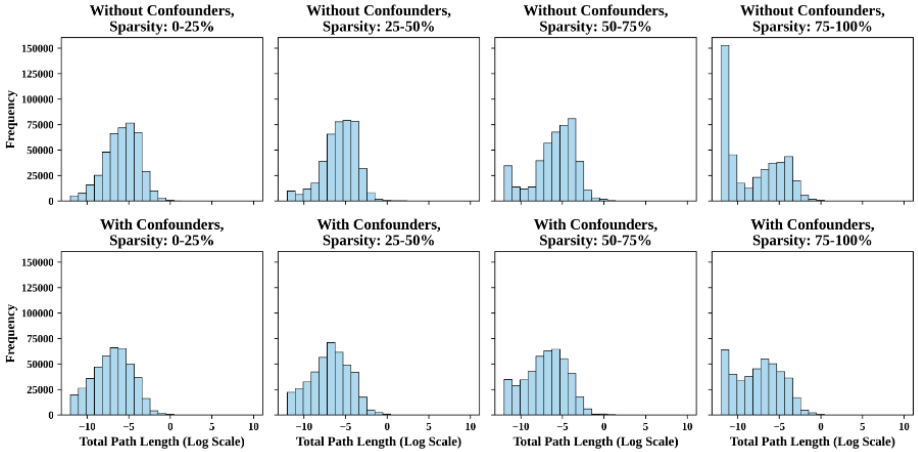


Combined results for $p = 4, 8, 12$.

Results: Computation Cost with Prior Knowledge

	Without Confounders				With Confounders			
Prior Knowledge p	0%	25%	50%	75%	0%	25%	50%	75%
4	9	9	8	6	9	9	8	6
8	35	34	27	18	39	40	37	22
12	107	99	81	42	184	198	174	74

Inspiration for Predicting Causal Graph Properties



Log-transformed Path Distributions for $p = 4, 5, 6$.

Predicting Using Path Distributions

- Use standardized moments to capture the shape of the distribution:

$$\tilde{\mu}_k = \frac{\mathrm{E}[(X-\mu)^k]}{(\mathrm{Var}[X-\mu])^{k/2}}.$$

- (Optional) Use ZDDs to enumerate path lengtths and compute moments. (see Lim et al. (Submitted))
- Use $p = 4, 5, 6$ for training and $p = 7$ to test.
- Create the following predictors:
  1. Confounder Detector - detect the presence of unmeasured confounders.
  2. Sparseness Estimator - estimate graph sparseness.
  3. Performance Predictors - predict the reliability of LiNGAM-SPP and DirectLiNGAM.

Predictor 1: Confounder Detector

Predict if unmeasured confounder is present.

Model	AUC	Precision	Recall	Accuracy	Optimal Threshold
*k*NN	0.6537	0.5918	0.7365	0.6142	0.4000
Random Forest	0.7608	0.6883	0.7465	0.7042	0.3300
XGBoost	0.7665	0.6955	0.7400	0.7080	0.3278
CatBoost	**0.7829**	**0.6957**	**0.7740**	**0.7178**	0.3271
AdaBoost	0.6997	0.6557	0.7540	0.6790	0.4972

Predictor 2: Sparseness Estimator

Predict if sparseness > 0.5.

Model	AUC	Precision	Recall	Accuracy	Optimal Threshold
*k*NN	0.6262	0.5949	0.6054	0.5905	0.4000
Random Forest	0.7286	0.7174	0.5640	0.6660	0.3400
XGBoost	0.7551	0.6773	**0.7443**	0.6902	0.2161
CatBoost	**0.7744**	0.7043	0.7217	**0.7050**	0.2647
AdaBoost	0.7579	**0.7285**	0.6438	0.6975	0.4958

Predictor 3: Performance Predictors

Predict if LiNGAM-SPP or DirectLiNGAM predicts the causal order correctly (i.e., $E_o = 0$).

	LiNGAM-SPP				DirectLiNGAM			
Model	AUC	Precision	Recall	Accuracy	AUC	Precision	Recall	Accuracy
*k*NN	0.641	0.155	0.613	0.617	0.638	0.147	0.594	0.615
Random Forest	0.822	0.272	0.752	0.768	**0.821**	0.239	**0.801**	0.725
XGBoost	0.799	0.237	**0.774**	0.721	0.800	**0.311**	0.648	**0.821**
CatBoost	**0.822**	**0.333**	0.698	**0.826**	**0.821**	0.306	0.711	0.809
AdaBoost	0.790	0.292	0.672	0.799	0.789	0.269	0.711	0.777

Conclusions

- Eliminated parameter tuning by replacing kNN-based estimators with PLR, improving performance and efficiency.
- PLR LiNGAM-SPP outperforms DirectLiNGAM in simulations and matches or exceeds its performance on real data.
- Incorporated prior knowledge with relative orderings, improving flexibility and performance.
- Developed confounder detector, sparsity estimator, and performance predictors using path distribution features, achieving high AUCs.

References I

Bach, F. R., & Jordan, M. I. (2003, mar). Kernel independent component analysis. *J. Mach. Learn. Res.*, *3*(null), 1–48. Retrieved from https://doi.org/10.1162/153244303768966085 doi: 10.1162/153244303768966085

Brooks, T., Pope, D., & Marcolini, M. (2014). *Airfoil Self-Noise.* UCI Machine Learning Repository. (DOI: https://doi.org/10.24432/C5VW2C)

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., & Reis, J. (2009). *Wine Quality.* UCI Machine Learning Repository. (DOI: https://doi.org/10.24432/C56S3T)

Gerritsma, J., Onnink, R., & Versluis, A. (2013). *Yacht Hydrodynamics.* UCI Machine Learning Repository. (DOI: https://doi.org/10.24432/C5XG7R)

Hyvärinen, A., & Smith, S. M. (2013). Pairwise likelihood ratios for estimation of non-gaussian structural equation models. *Journal of Machine Learning Research*, *14*(4), 111–152. Retrieved from http://jmlr.org/papers/v14/hyvarinen13a.html

References II

Ikeuchi, T., Ide, M., Zeng, Y., Maeda, T. N., & Shimizu, S. (2023). Python package for causal discovery based on lingam. *Journal of Machine Learning Research*, *24*(14), 1–8. Retrieved from http://jmlr.org/papers/v24/21-0321.html

Kalisch, M., & Bühlmann, P. (2007). Estimating high-dimensional directed acyclic graphs with the pc-algorithm. *Journal of Machine Learning Research*, *8*(22), 613–636. Retrieved from http://jmlr.org/papers/v8/kalisch07a.html

Lim, B. G., Tan, R. R., Kawahara, J., ichi Minato, S., & Ikeda, K. (Submitted). A recursive framework for evaluating moments using zero-suppressed binary decision diagrams. *IEEE Access*.

Nash, W., Sellers, T., Talbot, S., Cawthorn, A., & Ford, W. (1995). *Abalone.* UCI Machine Learning Repository. (DOI: https://doi.org/10.24432/C55C7W)

Pearl, J., & Mackenzie, D. (2018). *The book of why: the new science of cause and effect.* New York: Basic Books.

References III

Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D. A., & Nolan, G. P. (2005). Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, *308*(5721), 523-529. Retrieved from https://www.science.org/doi/abs/10.1126/science.1105809 doi: 10.1126/science.1105809

Shimizu, S. (2014, January). Lingam: Non-Gaussian Methods for Estimating Causal Structures. *Behaviormetrika*, *41*(1), 65–98. Retrieved 2022-11-04, from http://link.springer.com/10.2333/bhmk.41.65 doi: 10.2333/bhmk.41.65

Shimizu, S., Hoyer, P. O., Hyvarinen, A., & Kerminen, A. (2006). A Linear Non-Gaussian Acyclic Model for Causal Discovery. , 28.

Shimizu, S., Inazumi, T., Sogawa, Y., Hyvarinen, A., Kawahara, Y., Washio, T., … Bollen, K. (2011). DirectLiNGAM: A Direct Method for Learning a Linear Non-Gaussian Structural Equation Model. , 24.

References IV

Silva, R., Scheine, R., Glymour, C., & Spirtes, P. (2006). Learning the structure of linear latent variable models. *Journal of Machine Learning Research*, *7*(8), 191–246. Retrieved from http://jmlr.org/papers/v7/silva06a.html

Suzuki, J., & Yang, T. (2022). An Genenalized LiNGAM when confunder is present.

Suzuki, J., & Yang, T.-L. (2024). *Generalization of lingam that allows confounding.*