A Quantitative Causal Analysis for Network Log Data

Richard Jarry
Grenoble INP Ensimag
richard.jarry@grenoble-inp.org

Satoru Kobayashi
NII
sat@nii.ac.jp

Kensuke Fukuda
NII/Sokendai
kensuke@nii.ac.jp

***Abstract*—Data logs from network devices are primary data to understand the current status of operational networks. However, since many and heterogeneous devices generate network logs, extracting information on the network status from such logs is not an easy task in network operation, e.g., root cause analysis of network events. Though multi-variate time-series based log analyses extract correlation structure of the logs, identifying causality of the network logs is still a complex and challenging problem. The state of the art algorithm called the PC algorithm had been applied to network log analysis, but it has two fundamental limitations; (1) Generated graphs still have many undirected edges, and (2) Edges have no weight (whether plausible causality or not). To overcome these two limitations, in this paper, we rely on MixedLiNGAM to network log analysis; This algorithm produces weighted DAGs from a set of multi-variate log time series. In order to show the effectiveness of the proposed method, we apply MixedLiNGAM to a set of syslog data collected at a research and education network in Japan, and then compare output causal graphs generated by MixedLiNGAM and the PC algorithm. Our result demonstrates that obtained weighted directional edges help better understand the root cause of the network events.**

***Index Terms*—Log analysis, Network logs, Causal inference**

I. Introduction

Network log analysis is a fundamental and crucial task for network operation. However, a huge number and wide variety of log data can make log analysis difficult. For automating the network log analysis in this context, several topics have been well-studied such as anomaly detection [1, 2, 3, 4] and root-cause analysis [5, 6, 7] with mathematical/statistic methods.

To apply the mathematical methods, we need to convert raw log data into a set of event time-series per device per log type (i.e., log template) (Figure 1). Template generation algorithms extract log templates from raw log data. After matching the raw log data with the templates, we aggregate the number of log template appearances in a given time bin, and then obtain a log time-series per device per log template. Once generating a set of time-series, we can apply sophisticated mathematical methods.

One of the crucial properties in root cause analysis is *causality* of network events; it intuitively means whether log A causes log B. A straightforward approach to extract the causality is to use timestamp information in logs and domain knowledge. Decision tree-based approach is also applied in order to find temporal relations. Furthermore, several methods discover the causality from a set of time-series data with statistical metrics (e.g., Pearson correlation, confidence score, transfer entropy). However, these approaches have a common flaw; they cannot distinguish causality from co-occurrence (i.e., correlation) in a theoretical sense; thus they may produce spurious causal edges.

For solving this problem, causal inference techniques have been studied in several research fields [8]. Kobayashi et al. [5] apply a causal inference algorithm called PC algorithm [9] to network log data for the root cause analysis. The PC algorithm generates directed acyclic graphs (DAG) where a node is a log event, and an edge is the causality of log events. They demonstrate how the causal inference helps in detecting root causes of network trouble events. However, the PC algorithm has two significant drawbacks to apply to network log data. (1) The number of directed edges is relatively small due to a heuristic edge decision algorithm. In other words, the ability to discover causality is not so high in the PC algorithm. (2) Directed edges are unweighted; thus it is hard to discuss the likelihood of causal impact.

In this paper, we leverage a more sophisticated causal inference algorithm called MixedLiNGAM [10] for the network log data to overcome these limitations. MixedLiNGAM is based on skew-acyclic causal discovery methods, using data distribution shapes in order to infer causal directions (§ III). The advantage of this algorithm is that output DAG is fundamentally all directed, which enables us to weight edges in the DAGs. Thus, the result is more reliable and comparative than that in the PC algorithm. We apply MixedLiNGAM to a set of log data measured at a nationwide academic network in Japan (SINET) [11] to compare the effectiveness of two causal inference algorithms for network log analysis.

The main findings of the paper are as follows: (1) MixedLiNGAM discovers 10 times more causal edges than the PC algorithm (§ IV-B). Obtained weight values are high (ave 0.8), meaning that MixedLiNGAM discovers more reliable causal edges. (2) Our case study shows a weighted causal graph provides a more natural interpretation of the root cause that happened in the network (§ IV-C). (3) The overhead of processing time and memory usage is acceptable, compared with the benefits of the weighted DAG (§ IV-D).

II. Related works

Root cause analysis for network events has been intensively studied in the past. The past literature can be broadly classified

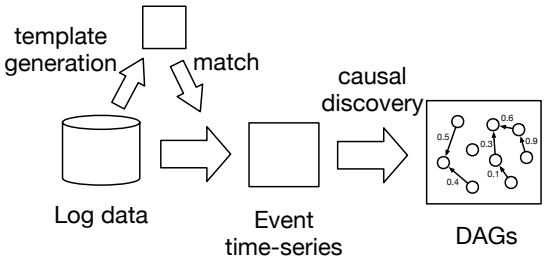

Fig. 1. Overview of causal inference in network log analysis

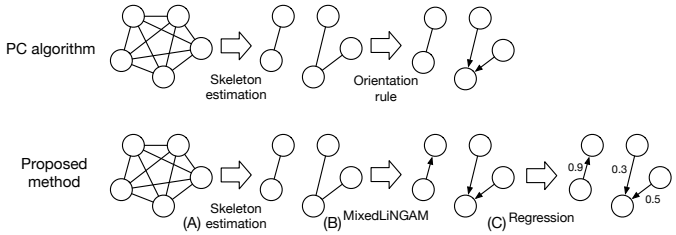

Fig. 2. Proposed method

into three approaches.

a) rule-based approach: is based on heuristic rules based on operators' domain knowledge. Timestamps and some important variables in log messages are used in building heuristic rules in Hadoop [12], Spark [13, 14], or cloud [15] environments. This approach is effective in a relatively small system (i.e., Hadoop), but it is difficult to generalize to other systems like heterogeneous network environments. Also, the accuracy of timestamps are not guaranteed in distributed systems, so detected causality may include pseudo causality (i.e., correlation).

b) decision-tree-based approach: mines relationships of multiple logs for the pinpoint root cause of events. Several decision-tree-based algorithms are applied to construct dependency graphs to find causality in log messages[16, 17, 18]. The main drawback of this approach is that large-scale log data is required to mine dependencies.

c) relation-mining approach: characterizes relationship between two log time series with statistical metrics, e.g., Pearson correlation [19], confidence score [20], transfer entropy [21]. This approach prunes unrelated edges in the initial complete graph depending on the value of the metrics. However, these methods may detect spurious causality because the metrics do not consider causality in a theoretical sense.

Ref. [5] and this work are categorized into the relation-mining approach. The advantage of the causal approach is to rely on a theoretical causality metric rather than the spurious ones, thus the causal approach could naturally remove the effect of correlation or co-occurrence.

III. Methodology

The overview of causal analysis is shown in Figure 1. This analysis flow basically follows that of Kobayashi et al. [5].

The input data is a set of syslog-like log messages for a time period (one day in this paper), including timestamp, source hostname, and free-format message. First, we generate log templates of the log messages with some template generation algorithms to aggregate the messages into a set of event time-series in a time bin (we use 60 seconds). We define a node of causal discovery as an event time-series, which corresponds to one per device per log type (log template). We also apply a time-series preprocessing to remove periodicity and regularity that causes detecting much false causality [5]. Finally, we conduct causal discovery with the event time-series input.

The causal discovery process of the proposed method is illustrated in Figure 2. This method consists of three steps: partially using PC algorithm to determine graph skeleton (i.e., no directions of the causal edges), estimating causal structure (i.e., directions) with MixedLiNGAM, and calculating the causal effect (i.e., weight) of the edges.

A. PC algorithm

Here, we introduce the PC algorithm [9], a constraint-based causal discovery algorithm. This algorithm starts with a complete graph of all nodes, and estimates the causal graph with following two steps (shown in Figure 2). First, the PC algorithm estimates graph skeleton by pruning non-causal edges on the basis of conditional independence [8]. Some conditional independence tests are used in this step (we use G square test [22]). Next, the PC algorithm determines directions of edges in the graph skeleton with orientation rule [23].

The orientation rule has a problem that not all edges can be directed with it because it works as trying to satisfy the definition of DAG. The output of the PC algorithm is in distinction with complete DAG and called CPDAG (completed partially DAG). This problem prevents us from calculating the causal effect of edges (details in § III-C).

B. MixedLiNGAM

To overcome this issue, we use MixedLiNGAM [10] to determine directions of edges. MixedLiNGAM is a hybrid causal model designed for processing both continuous and discrete data at the same time. The main idea behind MixedLiNGAM resolution algorithm is (1) to generate candidate DAGs and (2) to select the highest scored DAG from them.

The first step, generating candidate DAGs, relies on the concept of Markov equivalence classes. Markov equivalence classes are lists of equivalent DAGs having the same constraint (i.e., satisfying input graph structure). For example, if a graph skeleton is given as the input of MixedLiNGAM, the Markov equivalence classes are the possible combinations of directions of edges in the skeleton.

The next step, scoring DAGs, depends on the input data format (i.e., continuous or discrete). MixedLiNGAM uses the Bayesian information criterion (BIC) score [24], which represents the local log-likelihood of the causal structure. If the input data is continuous, the score is calculated on the basis of LiNGAM [25], a causal inference method for non-Gaussian data. If the input is discrete, the score is calculated with logistic regression.

Finally, we obtain the appropriate DAG structure with the highest score. All of the edges in the obtained DAG are directed by MixedLiNGAM.

To adapt the MixedLiNGAM to the network log data, we further extend the original MixedLiNGAM as follows: (1) To reduce the processing time for MixedLiNGAM, we separately apply MixedLiNGAM to the connected subgraphs in our implementation. This step would not change the combined results, but decrease the number of candidate DAGs (i.e., combinations of edge directions). (2) Theoretically, MixedLiNGAM can also start with CPDAG estimated by the full PC algorithm with orientation rule. This change will be a trade-off of improved processing time and lower reliability; the partially directed input will decrease the processing time of MixedLiNGAM, but the edge directions estimated with orientation rule is less reliable than that with MixedLiNGAM. In this paper, we do not use the orientation rule before MixedLiNGAM because the processing time is still reasonable without it (see § IV-D).

C. Regression

For calculating the weight of edges in a causal graph, the edges need to be all directed. The reason can be explained with the idea of Backdoor criterion [8]; a regression of causal effect (i.e., the weight of edges) should be calculated with an objective variable of a node and explanatory variables of the parent nodes, and no other nodes (e.g., child nodes) should not be included in the explanatory variables. If the explanatory variables include extra nodes, it causes false confounding and derives the wrong calculation of causal effect. Undirected edges in CPDAG prevent us from determining the parent nodes for regression. Therefore, we need all directed DAG for weighting the causal graph, and in our proposed method it is achieved with MixedLiNGAM.

MixedLiNGAM has two available regression methods in this step. For continuous input, it simply uses linear regression. However, for discrete (or binary) input, linear regression is not effective because it cannot consider probabilistic relations. Instead, MixedLiNGAM uses logistic regression for discrete (or binary) input. In the both cases, the calculated weight (regression coefficient) w represents the amount of dependency (e.g, if $w = 0.8$ on time-series events, an event occurrence increases the appearance of another event 0.8 times in average). MixedLiNGAM selects one of these regression methods for this step depending on the input data. The evaluation in § IV focuses on discrete input case.

D. Dataset

To show the effectiveness of MixedLiNGAM for network log data, we use a set of syslog data collected at a research and education network in Japan (SINET4) [11]. SINET4 connects over 800 universities/colleges/laboratories in Japan, and its backbone consists of 8 core routers and over 100 L2 switches. The raw data is 35M lines long and spans over 456 days from 2012 to 2013. Similar to the past literature [5], we construct log template time-series from raw log data using a supervised log template generation method (CRF). In the end, we pick up 30 days of data consisting of 8,605 time-series (i.e., nodes) from 128 devices for applying MixedLiNGAM in this paper.

IV. Analysis results

We first investigate the validity of MixedLiNGAM with synthetic time series data. Next, we focus on the effectiveness of the proposed approach in the log analysis by comparing macroscopic graph structures (especially, direction and weight of edges) generated by MixedLiNGAM and the PC algorithm (core algorithm of the state-of-the-art log causal analysis methods [5, 26]). Then, we demonstrate one case study to show how MixedLiNGAM works more intuitively than the PC algorithm in the context of network log analysis. Finally, we discuss the scalability of our MixedLiNGAM implementation by evaluating the processing time and memory usage.

A. Validation with synthetic data

Though the original MixedLiNGAM paper [10] evaluated their method with synthetic data based on the logistic model for discrete-continuous mixed data, our log time-series data does not clearly follow this model. Here, we validate MixedLiNGAM with more realistic synthetic data closer to real network log data.

Suppose that event occurrence can be approximated by Poisson processes. We consider causality as a probabilistic state transition; if A has a causality to B with $w = 0.8$, it means event A has 80% chance of triggering event B. Under this assumption, we generate a random causal DAG and its corresponding synthetic data that follows causal relations in the DAG. As in the evaluation parameters in the original MixedLiNGAM paper, the generated DAG consists of four data nodes, and there are 50% chances of causal edge existence between each node pairs. In the DAG, the causality weight w is a uniform random number where $0.5 \leq w < 1.0$. We generated a set of synthetic time-series data of 1-day or 7-days length with 1-minute bins (i.e., data size is 1,440 and 10,800, respectively) that follows the generated causality of the DAG. The generated time-series of an event is the mixture of two components; individual component of Poisson processes (control parameter λ, indicating how many times the events appear in one day on average) and external component of direct probabilistic causal effects from other connected nodes in the DAG. These validation settings can be reproduced with our open-source Python library causaltestdata [27].

With the random graphs and their synthetic time-series, we compared the accuracy of the two methods for restoring the causal structure. Table I shows the average accuracy of the methods with 100 different random DAGs. There are three metrics in the table: Skeleton accuracy is the accuracy in detecting correct skeleton edges (i.e., node pairs with some causality), Direction ratio is the ratio of correctly directed edges in the correct skeleton edges, and Weight diff. is the average difference of edge weight values in the correctly directed edges. Skeleton accuracy values of the PC algorithm and MixedLiNGAM are always the same because our

TABLE I
ACCURACY COMPARISON OF ESTIMATED CAUSALITY WITH PSEUDO DATA.

Method	Data model Size	λ	Skeleton accuracy	Direction ratio	Weight diff.
PC algorithm	1,440	10	0.878	0.170	–
	1,440	100	0.980	0.272	–
	1,440	1,000	0.993	0.211	–
	10,800	10	0.973	0.271	–
	10,800	100	0.993	0.270	–
	10,800	1,000	0.957	0.283	–
MixedLiNGAM	1,440	10	0.878	0.704	0.198
	1,440	100	0.980	0.651	0.124
	1,440	1,000	0.993	0.296	0.080
	10,800	10	0.973	0.768	0.087
	10,800	100	0.993	0.682	0.097
	10,800	1,000	0.957	0.240	0.242

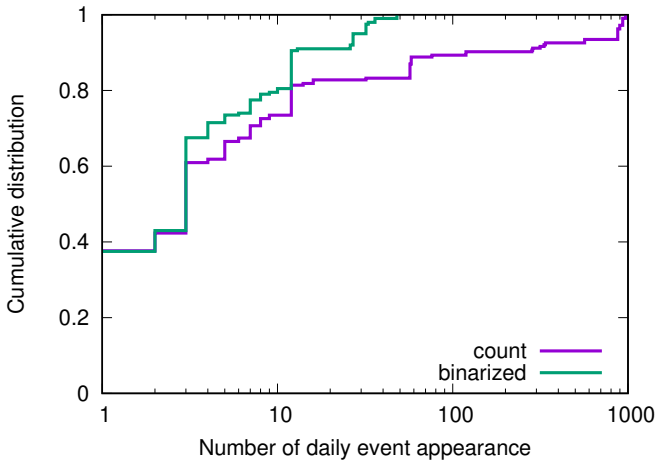

Fig. 3. Cumulative distribution of event appearance in case study data

MixedLiNGAM uses the PC algorithm to estimate the skeleton graph. In the evaluation of the original MixedLiNGAM paper [10], the Skeleton accuracy of the PC algorithm is always lower than 80% in the logistic model data. In contrast, the PC algorithm is more accurate for time-series data in our results. One possible reason is the difference in Conditional Independence tests used in the PC algorithm: original paper uses Fisher-Z test, and we use G square test. We had already confirmed that G square test is more appropriate than Fisher-Z test at least for network log analysis [5].

In Table I, the Direction ratio of MixedLiNGAM is larger than that of the PC algorithm. The direction ratio of the PC algorithm is lower than 0.3, which is not reliable for further analysis. This is because the PC algorithm cannot determine directions of all edges (see § III-A). In comparison, the direction ratio of MixedLiNGAM is improved to about 0.7 for most models. However, MixedLiNGAM is not accurate for the model with $\lambda = 1000$ (1,000 times per day). This inaccuracy comes from the LiNGAM's assumption: the data is non-Gaussian. If the time-series is too dense, the distribution gets close to Gaussian, causing accuracy degradation. Still, in the log analysis, the important log events (especially errors and warnings) are not so frequently appeared. Figure 3 shows the distribution of event appearance counts per day in the 1-day log data used in case study (§ IV-C). The PC algorithm uses binarized data, and MixedLiNGAM (i.e., direction part) uses the original count data. More than 90% of log events appeared less than 100 times per day (i.e., $\lambda \leq 100$). The mean of the counts per day is 79.4, and the median is 3 in this data. For all of the dataset explained in § III-D, the average of means is 70.8 and that of medians is 2.7. Therefore, our real case study data is slightly dense but does not largely differ from other days in our whole dataset.

B. Macroscopic analysis

TABLE II
MACROSCOPIC RESULTS (30 DAYS, 8605 NODES IN TOTAL)

Algorithm	#edges	#directed edges	ave. weight	stdev
Original PC	1289	121	–	–
MixedLingam	1289	1240	0.856	0.248

Here, we focus on the macroscopic graph structures obtained by the PC algorithm and MixedLiNGAM.

Table II lists the characteristics of the output graphs by the two algorithms. The graphs consist of 1075 non-isolated nodes and 1289 edges. Most of the nodes have no edges (i.e., isolated) after the edge pruning process. This is not surprising because those logs often indicate warnings or non-fatal errors, and therefore may not propagate. The numbers of nodes and edges are the same for the two algorithms because the edge pruning process is commonly applied.

Same as the validation results with synthetic data (§ IV-A), MixedLiNGAM determines much more directions of edges than the PC algorithm. The missing 49 edges in the MixedLiNGAM result are due to direction flapping, which is uncertainty in the most likely direction. This flapping happens rarely, generally in the smallest graphs when there are not enough variables for MixedLiNGAM to get conclusive results.

Furthermore, focusing on the average value of the weight w, we find that the obtained directed edges indicate significant confidence (e.g., close to 1.0). Thus, the causality of the output directed edges are more reliable than that obtained from the PC algorithm. Figure 4 represents the cumulative distribution of weight value of edges in MixedLiNGAM. We confirm that many of the edges have a high value of weight; For example, 60% of the edges for $w = 1.0$, 90% of the edges for $w \leq 0.5$. Thus, we conclude that MixedLiNGAM reliably and effectively extracts causal impacts in the network logs.

C. Case study

Next, we illustrate the usefulness of MixedLiNGAM output for network operation. We focus on one case study related to a link failure involving two connected devices (Router 1 and Switch 1). To discover the causality in this event, the PC algorithm output five nodes and four undirected and unweighted edges as shown in Figure 5 (a). Actual log messages corresponding to the IDs are listed in Table III with the timestamp. In this example, we see the templates related to

TABLE III
CAUSAL LOGS DURING LINKDOWN EVENT

TID	Timestamp	Log
43	16:22:04	R1: bgp_send: sending 21 bytes to 10.0.0.1 blocked: Resource temporarily unavailable
55	09:01:04	R1: bgp_hold_timeout:3814: NOTIFICATION sent to 10.0.0.1 (External AS X): code 4 (Hold Timer Expired Error: holdtime expired)
56	09:01:09	R1: RPD_BGP_NEIGHBOR_STATE_CHANGED: BGP peer changed state from Established to Idle
107	15:16:10	S1: EVT E4 PORT GigabitEthernet4/1 25011001 1350:AAAA Port up.
108	09:00:53	S1: EVT E4 PORT GigabitEthernet4/1 25011101 1350:AAAA Error detected on the port.

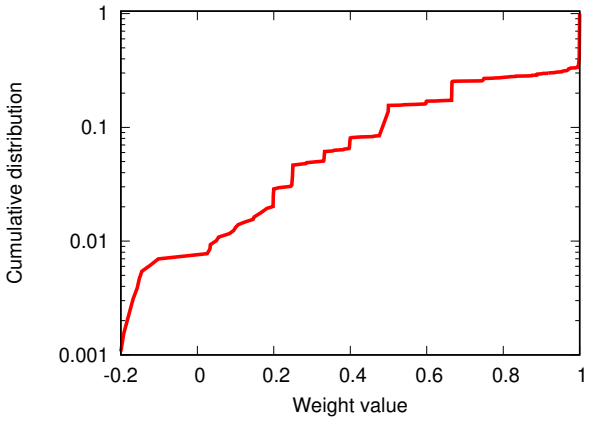

Fig. 4. Cumulative distribution of weight of edges

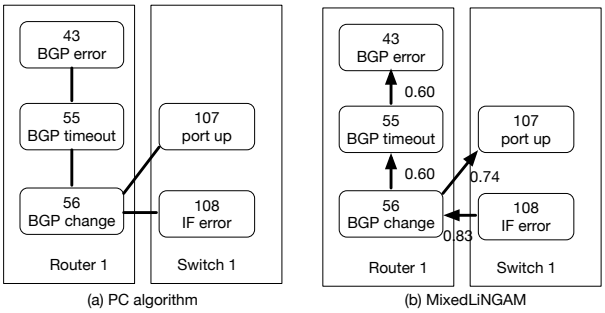

Fig. 5. Case study: Link down between two devices

interface and BGP. However, it is difficult to find any causality from this graph because no directed edges are discovered. This is consistent with the result on a few directed edges with the PC algorithm in the previous macroscopic characteristics.

On the other hand, Figure 5 (b) represents the graph structure obtained by MixedLiNGAM. It is clearly more informative and valuable for better understanding due to its directed and weighted edges. Furthermore, the edge direction is intuitively realistic; An interface error (ID 108) at Switch 1 triggers BGP state change (ID 56) at Router 1. Then, BGP process is timeout (ID 55) due to the link down. Furthermore, this BGP state change (ID 56) finally relates to port up (ID 107) at Switch 1 when the link is up again. Checking the timestamps in Table III, we notice that the link down and state change happen within 15 seconds, but the state change and port up are 6 hours apart. Thus, the causal inference algorithms are robust against two logs even temporally far from each other.

D. Performance evaluation

Finally, we evaluate the performance of the PC algorithm and our MixedLiNGAM implementation in processing time and memory usage. For the evaluation, we use a commodity computer (CPU: Intel Xeon E5-2680, Memory: 30GB).

Figure 6 displays the processing time overhead for the PC algorithm and MixedLiNGAM. The x-axis indicates the number of non-isolated nodes in the skeleton graph, and the y-axis is the total processing time in second. We confirm that the PC algorithm shows a low overhead due to the simplicity of the orientation rules. On the other hand, MixedLiNGAM is 2.4 times slower in maximum than the PC algorithm. However, considering the benefit of weighted directed graph output, we believe that this overhead is acceptable.

Figure 7 represents the memory usage of the two algorithms for different numbers of non-isolated nodes. We see that the overhead of the MixedLiNGAM and regression process is limited, compared with the edge pruning process.

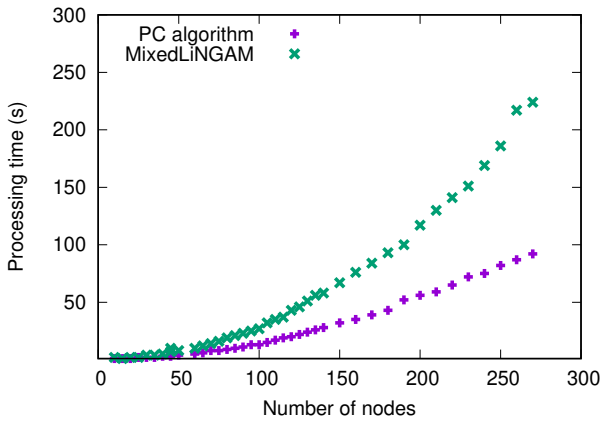

Fig. 6. Processing time overhead

V. CONCLUSION AND FUTURE WORK

In this paper, we introduced a causal inference algorithm for network log analysis. The key feature of the algorithm called MixedLiNGAM is to output weighted directed graphs from input log time-series data, though prior PC-algorithm based algorithms treat unweighted graphs. This weighted graph is expected to be more helpful in network root cause analysis. Our preliminary evaluation of MixedLiNGAM with actual network log data obtained at the nation-wide research and education network in Japan demonstrates that MixedLiNGAM outputs ten times more directed edges than the PC algorithm, and

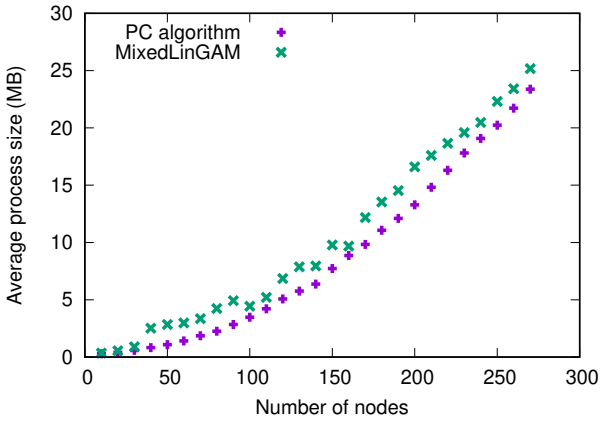

Fig. 7. Memory usage

the distribution of their weights indicates that MixedLiNGAM successfully extracts more reliable causal effects in the log analysis. Our primary contribution is to provide new insight into quantitative network root cause analysis.

We will investigate more characteristics of obtained weighted graphs toward more quantitative root cause analyses. We also intend to study more performance improvement of MixedLiNGAM to apply it to large-scale network data.

ACKNOWLEDGEMENTS

The authors thank Shohei Shimizu for providing us the original MixedLiNGAM implementation. This work is supported by the NII internship program and the MIC/SCOPE #191603009.

REFERENCES

[1] K. Otomo, S. Kobayashi, K. Fukuda, and H. Esaki, "Latent variable based anomaly detection in network system logs," *IEICE Transactions on Information and Systems*, vol. E102.D, no. 9, pp. 1644–1652, 2019.

[2] T. Kimura, A. Watanabe, T. Toyono, and K. Ishibashi, "Proactive Failure Detection Learning Generation Patterns of Large-scale Network Logs," in *Proceedings of IFIP/IEEE CNSM'15*, 2015, pp. 8–14.

[3] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the ACM SIGSAC CCS '17*, 2017, pp. 1285–1298.

[4] N. Aussel, Y. Petetin, and S. Chabridon, "Improving performances of log mining for anomaly prediction through NLP-based log parsing," in *Proceedings of IEEE MASCOTS '18*, 2018, pp. 237–243.

[5] S. Kobayashi, K. Otomo, K. Fukuda, and H. Esaki, "Mining Causality of Network Events in Log Data," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 53–67, Mar. 2018.

[6] Z. Zheng, L. Yu, Z. Lan, and T. Jones, "3-Dimensional root cause diagnosis via co-analysis," in *Proceedings of the 9th international conference on Autonomic computing - ICAC '12*. New York, New York, USA: ACM Press, 2012, pp. 181–190.

[7] A. Messager, G. Parisis, I. Z. Kiss, R. Harper, P. Tee, and L. Berthouze, "Inferring Functional Connectivity From Time-Series of Events in Large Scale Network Deployments," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 857–870, 2019.

[8] J. Pearl, *Causality: Models, Reasoning, and Inference*, 2nd ed. Cambridge Press, 2009.

[9] P. Spirtes, C. Glymour, and R. Scheines, *Causation, Prediction and Search*. The MIT Press, 1996.

[10] C. Li and S. Shimizu, "Combining Linear Non-Gaussian Acyclic Model with Logistic Regression Model for Estimating Causal Structure from Mixed Continuous and Discrete Data," *arXiv*, Feb. 2018, ref: 1802.05889. [Online]. Available: http://arxiv.org/abs/1802.05889

[11] S. Urushidani, M. Aoki, K. Fukuda, S. Abe, M. Nakamura, M. Koibuchi, Y. Ji, and S. Yamada, "Highly available network design and resource management of sinet4," *Telecomm. Systems*, vol. 56, pp. 33–47, 2014.

[12] J.-G. Lou, Q. Fu, Y. Wang, and J. Li, "Mining dependency in distributed systems through unstructured logs analysis," in *ACM SIGOPS Operating Systems Review*, vol. 44, 2010, p. 91.

[13] S. Lu, B. B. Rao, X. Wei, B. Tak, L. Wang, and L. Wang, "Log-based Abnormal Task Detection and Root Cause Analysis for Spark," in *Proceedings of IEEE ICWS 2017*, 2017, pp. 389–396.

[14] Z. Jia, C. Shen, X. Yi, Y. Chen, T. Yu, and X. Guan, "Big-data analysis of multi-source logs for anomaly detection on network-based system," in *IEEE CASE 2018*, 2018, pp. 1136–1141.

[15] B. C. Tak, S. Tao, L. Yang, C. Zhu, and Y. Ruan, "LOGAN: Problem Diagnosis in the Cloud Using Log-Based Reference Models," in *Proceedings of IEEE IC2E'16*, 2016, pp. 62–67.

[16] K. Nagaraj, C. Killian, and J. Neville, "Structured Comparative Analysis of Systems Logs to Diagnose Performance Problems," in *Proceedings of NSDI'12*, 2012, pp. 1–14.

[17] H. Yan, L. Breslau, Z. Ge, D. Massey, D. Pei, and J. Yates, "G-RCA: A generic root cause analysis platform for service quality management in large IP networks," *IEEE/ACM Transactions on Networking*, vol. 20, no. 6, pp. 1734–1747, 2012.

[18] J. Manuel, N. González, J. A. Jiménez, J. Carlos, D. López, and H. A. P. G, "Root Cause Analysis of Network Failures Using Machine Learning and Summarization Techniques," *IEEE Communications Magazine*, pp. 126–131, September 2017.

[19] E. Chuah, S.-h. Kuo, P. Hiew, W.-c. Tjhi, G. Lee, J. Hammond, M. T. Michalewicz, T. Hung, and J. C. Browne, "Diagnosing the Root-Causes of Failures from Cluster Log Files," in *IEEE HiPC 2010*, 2010, pp. 1–10.

[20] S. E. Solmaz, Bugra Gedik, H. Ferhatosmanoglu, S. Sözüer, E. Zeydan, and C. Ö. Etemoglu, "ALACA : A platform for dynamic alarm collection and alert," *International Journal of Network Management*, pp. 1–17, March 2017.

[21] V. Rodrigo, M. Chioua, T. Hagglund, and M. Hollender, "Causal analysis for alarm flood reduction," *IFAC-PapersOnLine*, vol. 49, no. 7, pp. 723–728, 2016.

[22] R. Neapolitan, *Learning Bayesian Networks*. Northeastern Illinois Universitya, 2004.

[23] T. Verma and P. Judea, "An Algorithm for Deciding if a Set of Observed Independencies Has a Causal Explanation," in *Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence*, 1992, pp. 323–330.

[24] G. Schwarz, "Estimating the dimension of a model," *The Annals of Statistics*, no. 2, pp. 461–464, 1978.

[25] S. Shimizu, P. O. Hoyer, A. Hyvarinen, and A. Kerminen, "A Linear Non-Gaussian Acyclic Model for Causal Discovery," *Journal of Machine Learning Research*, p. 28, 2006.

[26] S. Kobayashi, K. Otomo, and K. Fukuda, "Causal analysis of network logs with layered protocols and topology knowledge," in *Proceedings of CNSM'19*, 2019, pp. 1–9.

[27] "causaltestdata," https://github.com/cpflat/causaltestdata.