Comparative Causal Analysis of Network Log Data in Two Large ISPs

Satoru Kobayashi
NII
sat@nii.ac.jp

Keiichi Shima
IIJ
shima@wide.ad.jp

Kenjiro Cho
IIJ
kjc@iijlab.net

Osamu Akashi
NII
akashi@nii.ac.jp

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

***Abstract*—Towards a collaborative analysis of log data obtained from multiple networks, we first need to clarify what kind of information is available as transferable knowledge between different networks. However, we cannot directly compare network log data from different sources because the data largely depends on the network architecture and equipment. In this paper, we focus on relational information among network log events that follow standardized network protocols regardless of network environment. We propose a comparative analysis approach relying on causality between log time-series. In this approach, we classify log messages into anonymized log time-series with log templates, reduce the number of log time-series to decrease processing time, and apply causal discovery with the PC algorithm. To decrease the processing time of causal analysis, we propose a new preprocessing method that reduces the number of log time-series without any domain knowledge (i.e., available in any ISPs). We compare log data obtained from two nation-wide ISPs to demonstrate the effectiveness of the causal approach in comparative analysis.**

***Index Terms*—Network management, Log analysis, Causal discovery, Comparative analysis**

I. Introduction

Log data is an important data source for network management and troubleshooting. As its data volume vastly increases, many automated log analysis approaches have been proposed such as anomaly detection [1]–[3], fault localization [4], and root cause analysis [5]–[7].

Still, it is difficult to achieve automated network troubleshooting from log data. One reason for this difficulty comes from the lack of diversity in training data. Network troubles are caused by diverse factors in recent complicated systems. In addition, network operators usually do their best to prevent network troubles. If we only focus on the data of one network, the trouble data to be learned in machine learning approaches for automated troubleshooting is limited.

One possible approach for automated analysis based on richer knowledge is learning past troubles in the data of other networks. If we preliminary train an automated system with some valuable information from other networks, the system can provide useful information even for unseen troubles in the network. This collaborative system can be built on machine learning techniques such as transfer learning. However, it is unclear what knowledge can be transferred from different networks. Modern large-scale networks consist of complicated network technologies and their policies vary greatly from network to network. We first need to validate the network log data's similarity and difference as the input data source of future collaborative analysis for network troubleshooting.

978-1-6654-0601-7/22/$31.00

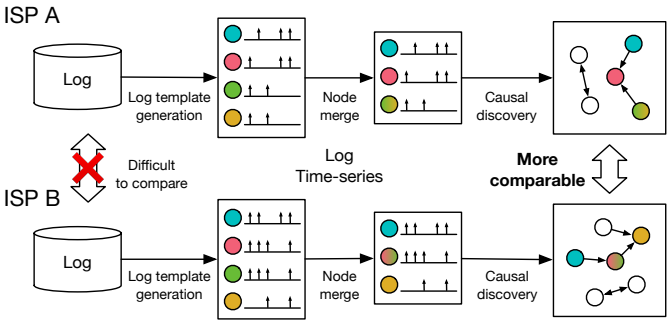

Fig. 1. Overview of comparative causal analysis

There are several challenges to comparing network log data obtained from different networks. Logically, the variables (i.e., IP addresses) appearing in the log message will differ in accordance with the networks. Also, as the deployed network devices are of different vendors or models, the log formats are different even if they correspond to the same behavior in the network protocols. Therefore, it is difficult to compare the logs in multiple networks directly. In addition, we need to consider the data publication policies. Network logs often include sensitive information such as private network configuration, network security policy, and sometimes customer activity.

In this paper, we propose a comparative analysis technique of network log data from different networks. The key idea is using causal analysis of log time-series events for the comparative analysis. A log time-series event is time-series data indicating the occurrence at each time bin of a log group corresponding to similar system behaviors. If there is a causality between given two log time-series events, the literal meaning is that one event causes the appearance of the other event. In this paper, we instead focus on the causality inferred from observed data, which corresponds to a direct relationship (or dependency) with a similar event appearance. This causal information is helpful to understand system behaviors that emit multiple log events together.

Figure 1 illustrates the abstracted analysis flow based on this idea. The log time-series is extracted from the classified log messages by log templates, which is a widely adopted

approach in past literature [4], [8]. The log time-series can be treated as the anonymized data of log messages whose variables (including sensitive parameters and addresses) are replaced with wildcards. Therefore, it is easier to share the anonymized log time-series than share log data by multiple network operators. Also, causal analysis extracts the relational information in the log time-series events (nodes in Figure 1). This relational information is more general and comparable than the raw log data or its time-series because the event dependency is usually derived from network protocols which is standardized in many cases regardless of device vendors. Log causal analysis is also effective for root cause analysis of network troubles as demonstrated in past literature [5].

It is also important to reduce the processing time in causal log analysis. A number of Internet service providers (ISPs) include too many network devices to analyze by conventional causal analysis. Past literature [6] proposed a preprocessing method based on domain knowledge, but this needs to manually format the domain knowledge, which is a time-consuming task for large ISPs. In this paper, we propose a new preprocessing method to merge synchronized time-series events, which reduces 52% of the processing time without domain knowledge. With the causal analysis, we apply a comparative causal analysis to real network log data collected in two nation-wide ISPs in Japan (ISP A and ISP B).

In this paper, we first explain the log causal analysis method including the new preprocessing method (section III). Then, we describe the overview of datasets (section IV) and validate our new preprocessing method (section V). We finally demonstrate the results of the comparative causal analysis with the log data of two ISPs (section VI).

The contributions of this paper are twofold. (1) We demonstrate that our causal approach is effective for comparative analysis of network logs obtained from different networks, and (2) we propose a new preprocessing method for log causal analysis and confirm that it can decrease the processing time and improve the reliability of causal analysis without domain knowledge of the network.

II. Related work

There are existing works that adopted causal analysis for system log analysis. Zheng et al. [9] use a causal approach to analyze log data of a supercomputer system. Zhang et al. [7] proposed a root cause analysis method for alarm data of a mobile network. Arya et al. [10] compared multiple causal analysis approaches with the log data of a benchmark microservice system. These works only focus on a single system or network.

In contrast, comparative analysis of log data has not been widely conducted in past literature. Oliner et al. [11] analyzed five supercomputer logs from the viewpoint of classification and correlation. Nagaraj et al. [12] analyzed the log data of three distributed applications with a number of approaches including a visualization based on the idea of dependency networks. These works are similar to our approach in that they consider relational information of log events (correlation and dependency). However, these existing works do not intend to compare network device logs from ISPs. In network log analysis, we should also consider the communication events between devices that follow common network protocols, so we need explicit relational information (i.e., causality).

III. Methodology

A. Overview

We describe the flow of comparative causal analysis in Figure 1. The processing flow follows that of our past literature [5], [6]: (1) generate log templates for input logs automatically, (2) generate time-series events (as input nodes for causal discovery) of classified log messages with the templates, (3) apply preprocessing methods to remove or decrease time-series events, and (4) use causal discovery to determine causal directed acyclic graph (DAG) structure between the remaining time-series event nodes.

The key idea of our causal analysis is removing spurious correlation with the PC algorithm [13]. The PC algorithm efficiently searches for conditional independence (i.e., spurious correlation) between input nodes. This algorithm is used in combination with a conditional independence test method such as the Fisher-Z test or G square test [14]. The input data of the PC algorithm is a data matrix of vectors corresponding to the nodes and the data format depends on the conditional independence test (e.g., discrete data for G square test). The PC algorithm is simple but fast compared with other approaches such as Granger causality [10], so the approach is suitable for analyzing large-scale data. With this algorithm, we estimate causal relations between time-series nodes of log events.

First, we need to generate log templates of the input logs to classify them as time-series. A log template is a format of the message part in syslog messages, which consists of description words (matching with the input messages) and variable wildcards. There are many log template generation methods [4], [8], [15], [16]. From the intensive accuracy comparison of the methods [17], we choose CRFe [17], a supervised learning method with higher accuracy than unsupervised methods, for the dataset with training data (i.e, the ground truth of log template generation), and Drain [8], a state-of-the-art unsupervised method, for the dataset without training data.

Next, we classify the log messages into time-series events with the generated templates. We use one-day-long time-series nodes as the input of causal analysis to focus more on the temporal behavior of the target network devices (i.e., we obtain one causal DAG for one-day-long log data). In this step, we consider a time-series node with a sequence of logs with a common log template and from a common host device. Here, we use amulog [17], [18], which can quickly classify the log messages with log templates automatically generated with multiple log template generation methods.

Then, we apply a number of preprocessing methods to modify the input time-series events. The methods are intended to make the causal analysis efficient and reliable. In our

past works, we had proposed a preprocessing method to decrease periodic or constant time-series [5] and a preprocessing method to use the domain knowledge of the network environment for reliable causal analysis [6]. In this paper, we additionally propose a new preprocessing method, discussed in subsection III-B, that can be used together with these existing preprocessing methods without any prior domain knowledge.

Finally, we apply the PC algorithm to the time-series nodes. To handle the sparse log time-series in causal analysis, we first transform the input time-series into binary values that correspond to the appearance of events in each one-minute time bin. We use the G square test [14], a conditional independence test for discrete input, in the PC algorithm. There are a number of variations of the PC algorithm, and we use the stable-PC algorithm [19], [20] that is order-independent to the input. The output of the PC algorithm is a completed partially directed acyclic graph (CPDAG, a potential DAG including undirected edges) of the time-series nodes. Therefore, we estimate one CPDAG from one-day log data obtained from one network.

This analysis flow also has an advantage in computing resource management. Steps (3) - (4) require node IDs and their time-series as the input (i.e., no log templates and hostnames, as far as not applying the preprocessing method based on domain knowledge). Therefore, we can use external computing resources for these steps without violating any data management policies because the input does not include sensitive information. We later match the output CPDAG with node specifications (i.e., log templates and hostnames) and then understand the obtained causal results. For example, we analyze log data of ISP A in external computation servers with this technique. The causal analysis approach (including the external computation) is available in logdag, our open-source causal analysis platform [21].

B. Preprocessing

Causal discovery methods including the PC algorithm require large processing time. The computational complexity of the PC algorithm is more than $O(n^2)$ where n is the number of input nodes. To analyze log data of large-scale networks, we need to consider efficient preprocessing methods to decrease the processing time of causal discovery.

We had proposed two preprocessing methods for log causal analysis. One method decreases periodic or constant components in the time-series [5]. If a number of time-series nodes have periodicity with the same intervals, they form false causality that cannot be removed as spurious correlation with causal inference (i.e., conditional independence test). This preprocessing method removes such periodicity from the time-series with Fourier analysis and linear regression. This method effectively decreases the number of input nodes, but we also need other preprocessing methods because it only focuses on periodic events.

The other method uses domain knowledge of the network (in particular, network topology and protocol layers of events) [6] in causal analysis. This method prunes causal edge candidates between the input nodes if the candidates are considered not reasonable on the basis of the domain knowledge. It is also effective for decreasing the processing time of the PC algorithm by reducing the number of calculations for the conditional independence test. However, we need to define the domain knowledge manually for this method (even for preliminary analysis), which is time-consuming even for well-experienced operators. In this paper, we could not use this preprocessing method for the dataset of ISP A.

Here, we propose a new preprocessing method: merging multiple nodes with completely synchronized time-series. The time-series nodes include a number of event pairs with completely synchronized time-series (i.e., the time-series are equal or integer multiplying). We consider these synchronized event pairs as one virtual event and merge the nodes into one node. By repeating the node merging, we obtain a compressed time-series input for causal discovery.

The idea of node merging comes from the sparseness of log time-series. For example, in our preliminary survey, the median of average time-series appearances is two in the log time-series of ISP B, which means half of the events only appear less than twice in a day (though the mean is 19). In this sparse data, many pairs of nodes have completely synchronized time-series. It is not against our intuition because a number of log events should strictly follow other events in normal situations. However, the stable-PC algorithm outputs unintuitive results among more than three nodes with the same time-series; Although these nodes have clear correlation and intuitively have causality or at least indirect causal relations, the algorithm removes all the edge candidates between the nodes because every edge candidate between a pair of nodes is always considered as spurious correlation given the other node. This behavior causes the algorithm to estimate shredded causal results which are ineffective for root cause analysis based on the estimated causal flows. Unfortunately, any causal inference methods cannot solve this problem because the reason comes not from the method but the input data.

The proposed preprocessing method has three advantages. First, the method does not require any additional data than the input time-series (i.e., using no domain knowledge). We can apply this method to any dataset or network environment. Second, it decreases processing time for testing conditional independence related to the same time-series. We discuss the effectiveness in section V. Third, it makes the causal results more reliable. The relations of nodes with synchronized time-series are represented as "same node" instead of causality or non-causality. It is valuable information that the relations are not deterministic in the input data.

In addition, this node-merging method can be used together with the other two methods. If one uses the node-merging method and domain-knowledge-based method together, the node merging needs to consider an additional constraint; the merged nodes need to be from the same host device and of the same layer in the domain knowledge. This is because the input nodes of the domain-knowledge-based method must be consistent in its host device and layer.

TABLE I
DATASETS

Network	#Hostnames	Period (day)	#Templates	#Log lines	#Tickets
ISP A	1,861	92	5,182	56,968,361	36
ISP B	131	365	1,789	34,722,785	88

TABLE II
VALIDATION OF NODE-MERGING METHOD WITH THE DATASET OF ISP B

Method	Time (sec)	#Nodes	#Edges	#Tickets
Conventional	76.0	360.1	56.8	70 (80%)
Node-merge	36.3	279.4	49.8	71 (81%)

IV. DATASET AND ANALYSIS SETUP

In this paper, we use network logs from two different ISP networks, ISP A and ISP B, for the comparative analysis. In this section, we introduce the two datasets and the methods to analyze them. Note that we use our best methods for each dataset (including different methods with each dataset) because we do not intend to compare their accuracy competitively but to compare their trends in the viewpoint of causality.

Table I shows the overview of these datasets. #Hostnames means the number of different hostnames in the syslog header. The number of hostnames in ISP A is much larger than that of ISP B, but the number includes multiple virtual routers and routing engines in a device. #Templates is the number of log templates generated with different methods: Drain [8] for ISP A and CRFe [17] for ISP B. This is because we have sufficient training data of ISP B for supervised learning, but not that of ISP A. In addition, we use partially different sets of methods in the preprocessing part of log causal analysis; we commonly use the periodicity-based method and node-merging method for both datasets, and we additionally use the domain-knowledge-based method only for ISP B.

As the ground truth to classify the network behavior found as the causality, we use trouble tickets recorded by the operators of each network. We focus on the tickets related to the target hostnames and of which troubles had lasted more than 10 minutes (described as "All Tickets" in section VI). #Tickets in Table I is the number of the focused trouble tickets. These ground truth data are used mainly in section VI.

V. VALIDATION

In this section, we demonstrate the effectiveness of the newly proposed preprocessing method based on node-merging for log causal analysis. We compare two causal analysis results with and without the node-merging method (Conventional and Node-merge, respectively) in the dataset of ISP B; we do not compare them in the dataset of ISP A because the causal analysis for ISP A does not end in a reasonable time, at least in two weeks, without the node-merging method. Table II shows the results. In the table, Time is the average processing time of causal discovery (including preprocessing) per day. We can see that our node merging method decreases 23% of nodes from the causal analysis input. It causes 52% reduction of processing time for causal discovery because the processing time depends on more than the square of the number of input nodes as explained in subsection III-B.

In addition, we can see that the node merging method does not degrade the reliability of causal analysis results. #Tickets is the number of trouble tickets that our system provides more than one corresponding edge (i.e., both end node events are related to the tickets). Both methods estimate causal edges corresponding to about 80% of tickets, though the node merging reduces 10% estimated causal edges. Therefore, we can conclude that the proposed method improves the performance and the reliability of the causal analysis.

TABLE III
CAUSAL ANALYSIS RESULTS OF TWO ISPS

Network	#Nodes	#Edges	#Tickets
ISP A	2,758.3	349.8	18 (42%)
ISP B	279.4	49.8	71 (81%)

TABLE IV
CLASSIFICATION OF TROUBLE TICKETS

Network	Class	#All tickets	#Tickets with edges
ISP A	Circuit	22	15 (68%)
	Connection	7	0 (0%)
	Device	7	3 (43%)
ISP B	Circuit	22	14 (63%)
	Connection	55	50 (91%)
	Device	7	4 (57%)
	Blackout	4	3 (75%)

VI. EVALUATION

Here we compare the causal results of ISP A and ISP B.

A. Classification of trouble tickets

First, we compare the quantitative aspects of the causal analysis results, as shown in Table III. "#Nodes" and "#Edges" are the average numbers in one output CPDAG respectively. We can see that the number of edges is much smaller than that of nodes in both networks. This means we require less effort for checking causal edges than for checking log events that are classified with log templates and hosts.

To understand the estimated causal edges in detail, we manually annotate the causal edges with related trouble tickets (mainly on the basis mainly of message descriptions and the locality of event time-series).

We show the overview of causal analysis results with the classification of tickets annotating the edges. Table IV shows the classification of trouble tickets corresponding to the estimated causal edges. The tickets are manually classified into four classes in the table. Circuit tickets correspond to the link down of circuits in the network. Connection tickets include connection errors (except link down) between the devices. Device tickets are related to errors that are confined to a single device. Blackout tickets come from scheduled blackouts, which is not our focus in this analysis.

Our method finds causal edges related to about two-thirds of the Circuit tickets, which is a common trend in ISP A and

TABLE V
CLASSIFICATION OF ADJACENT NODES OF EDGES RELATED TO CIRCUIT TICKETS

Network	Node label	Days w/ logs	Days w/ edges	Days w/ tickets (edges/tickets)
ISP A (92 days)	MPLS	88	69	12 (17%)
	System	92	92	5 (5%)
	Interface	92	92	5 (5%)
	Monitor	90	53	4 (4%)
	OSPF	61	5	1 (20%)
ISP B (365 days)	Monitor	191	60	10 (17%)
	MPLS	39	13	4 (31%)
	BGP	315	291	4 (1%)
	Interface	318	211	3 (1%)
	OSPF	54	1	1(100%)

TABLE VI
CLASSIFICATION OF EDGES RELATED TO CIRCUIT TICKETS

Network	Label 1	Label 2	Same host	Days w/ edges	Days w/ tickets (edges/tickets)
ISP A (92 days)	MPLS	MPLS	✓	11	5 (45%)
	System	System	✓	91	3 (3%)
	MPLS	MPLS		28	5 (18%)
	Monitor	Monitor		22	3 (14%)
	System	System		13	2 (15%)
	Interface	Interface		59	3 (5%)
	Monitor	OSPF		1	1 (100%)
	Interface	OSPF	✓	1	1 (100%)
	Interface	Monitor		1	1 (100%)
	System	MPLS		2	2 (100%)
	System	Interface		3	1 (33%)
	Interface	Monitor	✓	1	1 (100%)
	Monitor	Monitor	✓	1	1 (100%)
	Monitor	MPLS		1	1 (100%)
ISP B (365 days)	Monitor	Monitor	✓	28	9 (32%)
	BGP	BGP	✓	215	4 (2%)
	Monitor	MPLS	✓	5	4 (80%)
	Interface	Interface	✓	166	3 (2%)
	Monitor	BGP	✓	3	1 (33%)
	Monitor	MPLS		1	1 (100%)
	MPLS	MPLS	✓	1	1 (100%)
	MPLS	MPLS		3	1 (33%)
	OSPF	OSPF	✓	1	1 (100%)

ISP B. However, the trends related to Connection tickets are significantly different in ISP A and ISP B. This comes from the different policies to report trouble tickets between ISP A and ISP B. In our manual survey, Connection tickets in ISP B include many connection troubles caused by peering partners, which are not included in ISP A tickets. On Device tickets, the trends of the two networks are close numerically. Still, the included tickets are significantly different and disparate because ISP A and ISP B use different vendors or network device models.

B. Details in causal edges

Considering these findings, we next focus on Circuit tickets because they are more comparable than other classes in the different network environments. To understand the network events related to these tickets, we manually annotated these event nodes with six labels on the basis of their log templates. "System" events consist of hardware events and self-contained events. "Interface" events are related to network interfaces on the network devices. "Monitor" events include secondary events for monitoring or management such as SNMP, STP, and NTP. The other three labels ("BGP", "MPLS", "OSPF") correspond to the communicative events of their respective network protocol names.

Table V and Table VI demonstrate the detailed feature of edges annotated with Circuit tickets. In Table V, the values are aggregated with the adjacent nodes of the edges. In Table VI, the values are aggregated with the pair of end nodes and their host relations (same hostnames or different hostnames) of the edges (considering no direction of edges but node pairs only). In these tables, each row corresponds to a group of edges identified with the event labels and has multiple count values aggregated by days (i.e., the number of estimated CPDAGs). "Days w/ logs" (only in Table V) is the number of days with more than one corresponding log event appearance (not considering the causal analysis results). "Days w/ edges" is the number of days that include the same edges as ones related to the Circuit tickets. If "Days w/ edges" and "Days w/ logs" are equal, the events are always related to a number of other events (i.e., within a regular state transition). "Days w/ tickets" is the number of days corresponding to the edges related to Circuit tickets (listed with its ratio to "Days w/ edges" in percentage). If the ratio of "Days w/ tickets" to "Days w/ edges" is close to 100%, the edges only appear when there is a Circuit trouble. Therefore, we consider the groups with a large value of this ratio as important and valuable events in Circuit troubles.

We can see there are four common node labels in the two networks in Table V. "MPLS" nodes frequently appear in common with the tickets, but the specificity is different: these MPLS events are specific to Circuit troubles in ISP B, but not in ISP A. This is due to the usage difference of MPLS functions in these networks. We can also see that "OSPF" events rarely form causal edges though their logs appear frequently. These "OSPF" related edges are strongly tied to Circuit troubles in both networks, so they can be valuable information with Circuit troubles. Other node events regularly appear as adjacent nodes of edges.

In Table VI, most of the edges are between the same event labels (orange-colored). As the causal analysis does not focus on anomalies or outliers, the causal edges include normal (and sometimes self-evident) state changes. There are also many edges between different event labels (blue-colored), and we can see they are adjacent to Monitor or Interface event nodes in many cases. Monitor and Interface events are subordinate functions to the original network services, so they often have some dependency (i.e., causality) with other functions.

Figure 2 shows the causal edges related to a Circuit ticket in ISP B. In this ticket, there is a circuit break between two routers (routers A and B) in different datacenters. In Figure 2, we can see two edges of different event labels between different devices: MPLS (RSVP) to Monitor (SNMP) and BGP to Monitor (NTP). The event pair of MPLS and Monitor also exists in ISP A, so this is more important than the other one (BGP to Monitor) as transferable knowledge in future collaborative analysis.

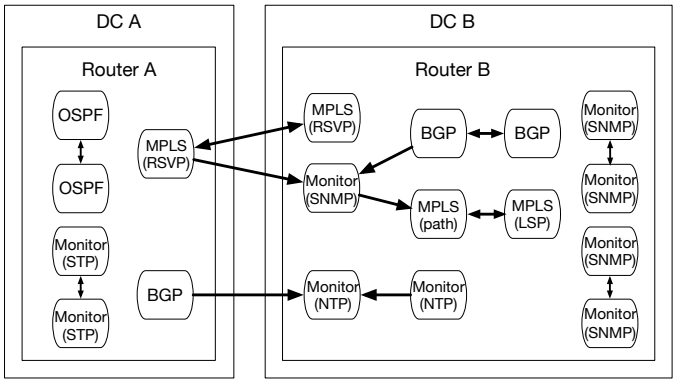

Fig. 2. Example of CPDAG corresponding to a Circuit ticket in ISP B

C. Summary and findings

Finally, we discuss the effectiveness of the causal approach in comparative analysis. As shown in subsection VI-A, we can quickly figure out the mutual behaviors in network devices by focusing on causal edges. In particular, focusing on Circuit troubles, the related log events regularly appear in Table V (see "Days w/ logs"), so it is difficult to identify the important log events for the troubles without relational approaches. In contrast, with causal edges, we can successfully find many types of event pairs specific to Circuit troubles in Table VI. We can compare valuable knowledge of different networks hidden in regular events with causal analysis.

In this evaluation, we focus on Circuit troubles. It is difficult to compare causality related to other trouble tickets (Connection and Device) in these logs. Connection tickets consist of connection errors without circuit breaks and they often include protocol-specific errors that largely depend on the network architecture. Device tickets are related to internal errors depending on the device's hardware or software. In the comparison of ISP A and ISP B, these conditions are significantly different. If one uses network log data from many more ISPs for collaborative analysis, these troubles can include useful information because a number of the ISPs consist of similar network architecture or devices.

VII. Conclusion

In this paper, we demonstrated a comparative analysis of log data obtained from two different ISPs. We proposed a causality approach for the comparative analysis that estimates causal relations between log time-series events. To decrease the processing time for causal analysis without any domain knowledge, we proposed a new preprocessing method for causal discovery that merges nodes of the synchronized time-series. The method successfully decreased more than half of the processing time in the log data of ISP B. We then investigated the obtained causal results with trouble tickets and demonstrated the effectiveness of the causal approach in comparative log analysis. Through the analysis, we demonstrated that the causal approach determines relational information hidden in straightforward log analysis.

As future work, we will focus on collaborative analysis approaches of network log data on the basis of findings from the comparative analysis in different networks. We will also explore measures for network operators to easily compare network logs or anonymized time-series events.

Acknowledgements

This work is supported by 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, 9 2019.

[2] W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun, and R. Zhou, "Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs," in *IJCAI International Joint Conference on Artificial Intelligence*, 2019, pp. 4739–4745.

[3] S. Huang, Y. Liu, C. Fung, R. He, and Y. Zhao, "HitAnomaly : Hierarchical Transformers for Anomaly Detection in System Log," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2064–2076, 2020.

[4] T. Kimura, K. Ishibashi, T. Mori, H. Sawada, T. Toyono, K. Nishimatsu, A. Watanabe, A. Shimoda, and K. Shiomoto, "Spatio-temporal factorization of log data for understanding network events," in *IEEE INFOCOM'14*, 2014, pp. 610–618.

[5] S. Kobayashi, K. Otomo, K. Fukuda, and H. Esaki, "Mining causes of network events in log data with causal inference," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 53–67, 2018.

[6] ——, "Causal analysis of network logs with layered protocols and topology knowledge," in *Proceedings of CNSM'19*, 2019, pp. 1–8.

[7] K. Zhang, M. Kalander, M. Zhou, X. Zhang, and J. Ye, "An Influence-Based Approach for Root Cause Alarm Discovery in Telecom," in *Proceedings of ICSOC 2020*. Springer, 2020, pp. 124–136.

[8] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An online log parsing approach with fixed depth tree," in *Proceedings of IEEE ICWS'17*. IEEE, 2017, pp. 33–40.

[9] Z. Zheng, L. Yu, Z. Lan, and T. Jones, "3-Dimensional root cause diagnosis via co-analysis," in *Proceedings of ICAC'12*, 2012, p. 181.

[10] V. Arya, K. Shanmugam, P. Aggarwal, Q. Wang, P. Mohapatra, and S. Nagar, "Evaluation of Causal Inference Techniques for AIOps," in *ACM SIGKDD CODS COMAD 2021*, 2021, pp. 188–192.

[11] A. Oliner and J. Stearley, "What Supercomputers Say: A Study of Five System Logs," in *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*. IEEE, 2007, pp. 575–584.

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

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

[14] R. E. Neapolitan, *Learning Bayesian Networks*. Prentice Hall, 2004.

[15] R. Vaarandi, "A data clustering algorithm for mining patterns from event logs," in *Proceedings of IPOM'03*, 2003, pp. 119–126.

[16] S. Kobayashi, K. Fukuda, and H. Esaki, "Towards an NLP-based log template generation algorithm for system log analysis," in *Proceedings of CFI'14*, 2014, pp. 1–4.

[17] S. Kobayashi, Y. Yamashiro, K. Otomo, and K. Fukuda, "amulog: A general log analysis framework for comparison and combination of diverse template generation methods," *International Journal of Network Management*, 2021, http://doi.org/10.1002/nem.2195.

[18] S. Kobayashi, Y. Yamashiro, K. Otomo, K. Fukuda, and H. Esaki, "amulog: A General Log Analysis Framework for Diverse Template Generation Methods," in *Proceedings of CNSM'20*, 2019, pp. 1–5.

[19] D. Colombo and M. H. Maathuis, "Order-independent constraint-based causal structure learning," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3741–3782, 2014.

[20] T. 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. 13, no. 9, pp. 1–13, 2014.

[21] "logdag," https://github.com/cpflat/logdag.