Learning Causal Knowledge Graphs from Text Log Data

Aleksandr Eismont

Karlsruhe Institute of Technology, Kaiserstr. 12, 76131 Karlsruhe, Germany
ukqln@student.kit.edu

Abstract. A fundamental task in various science disciplines is to find underlying causal relations and use them. Finding out why an event occurs and its cause means that we can, for example, stop the effect from happening if we remove this knowledge from the equation or generate the subsequent effect if we replicate it. A traditional way to discover causal relations is to use interventions or randomized experiments, which is, however, in many cases, too expensive, too time-consuming, unethical, or even impossible. As a result, causal knowledge can be derived from observational data in a purely data-driven manner. This paper addresses the challenges in estimating the causal generating processes for time-series data. It helps researchers in different fields, from its use in medical research to climate and cloud computing, among many others. This paper aims to provide an introduction and a brief review of the computational methods for causal discovery, including constraint-based, score-based methods, and those based on functional causal models.

Keywords: Conditional Independence · Causal Discovery · Network logs

1 Introduction

Almost all science is about identifying causal relations and the laws or regularities that govern them. Interventions or randomized experiments are a common technique to find causal relationships, but they are often too expensive, time-consuming, or even impossible to do. Therefore, researchers have drawn much attention by revealing causal information by analyzing purely observational data, a process known as causal discovery [31].

In automated analysis, causal discovery is a prominent approach to extracting contextual information. It removes spurious correlations from the results, allowing operators to concentrate on crucial details. One practical and natural method for efficient analysis is to leverage domain knowledge on a target network. Network operators empirically select valuable information from various data sources based on their domain knowledge in manual operation. This information selection is not always accurate, but it effectively improves troubleshooting.

Software systems are becoming increasingly massive and complex, containing hundreds of services distributed across thousands or even hundreds of thousands of servers and supporting many concurrent users. One particular challenge for large-scale software systems is anomaly diagnosis. When problems occur, how can they be quickly diagnosed, and how can administrators rapidly identify root causes? Logs are a straightforward and common source of information for problem diagnosis [2,8,26]. Typically, administrators manually check log files and search for problem-related log lines. However, in today's large-scale systems, logs can be overwhelmingly large. For instance, daily log data could reach tens of terabytes in some large-scale systems that provide global services. On the other hand, problems of today's system's architectures can be cross-component and cross-service; for example, it is hard to get root causes based on particular "error" logs. As a result, identifying problems by hand can be time-consuming and error-prone. Understanding the dependencies among different components of a large-scale distributed system is extremely important for problem diagnosis.

This paper aims to provide a comprehensive review of learning causality from massive log data. Below, I present an outline of the topics covered in this paper. First, in Section 2, I introduce a general approach structure that I have identified in my literature study on what methods exist. Section 3 focuses on the methods developed for the problem of efficient log template generation. Section 4 discusses the widely used straightforward methods for event time-series creation. Afterward, in Section 5, I discuss the preliminaries of learning causality from data for causal discovery. I will start with the constraint-based, score-based, and functional-based methods. At the end of this section, I introduce one of the most known algorithms to deal with causal, temporal data — the Granger causality. Finally, Section 6 gives a brief conclusion.

2 Approach Structure

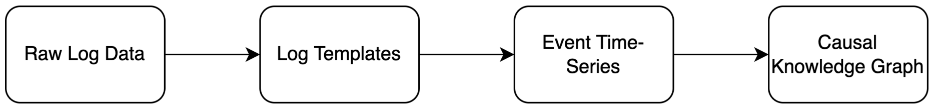


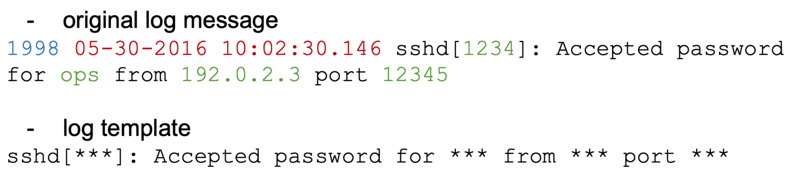
Fig. 1. The general processing flow.

The overview of causal analysis literature is shown in Figure 1. This analysis flow follows the main existing approaches [17,18,19,21,26].

The input data is a set of raw log messages, including timestamps, source parameters, and free-format messages. 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 define a node of causal discovery as an event time-series, corresponding to one per device per log type (log template). Some approaches also apply time-series preprocessing to remove periodicity and regularity, which causes false causality detection [17,19]. Finally, we conduct causal discovery with the event time-series input to get the causal knowledge graph.

3 Log Templates Generation

One challenge must be carefully considered when applying causality discovery algorithms to log data. Raw log messages cannot be analyzed directly with sta-



- original log message

```
1998 05-30-2016 10:02:30.146 sshd[1234]: Accepted password
for ops from 192.0.2.3 port 12345
```

- log template

```
sshd[***]: Accepted password for *** from *** port ***
```

Fig. 2. Example of log template.

tistical approaches because they are strings, not numerics. The content of a log record is an unstructured free-text written by software developers, which makes it challenging to structure. Log messages often contain two types of information: a free-form text string used to describe a recorded program event and parameters used to express some essential characteristics of the current task. The log template is a log format, the variables of which are replaced by parameter placeholder *. So, we need to extract log templates from the original log messages except for IDs and timestamps. Figure 2 is an example of a log message and the corresponding log template. In this example, variables such as process ID, user name, IP address, and port number are replaced by stars.

However, Kobayashi et al. [17,19] consider one more issue that appears by log messages generation. The distribution of log appearances is long-tailed; like daily processes, some events occur much more frequently than others, like error logs. Such a periodic log template represents regular daily events caused by an event timer. This difference in frequency significantly affects the algorithms. Indeed, the algorithms detect more edges from frequent events while hiding important relations of minor events. To avoid this, they propose to remove events that appear frequently but are less critical in troubleshooting. For this purpose, the authors use a Fourier analysis to find periodic events of large intervals and a linear regression analysis for short intervals.

Traditional log parsing techniques rely on human experts' regular expressions designed and maintained. Large systems with various software and hardware components render it intricate to support this manual effort. Thus, automated log parsing is essential due to its practical relevance to maintenance systems. Parsing techniques can be distinguished in the following classes based on various aspects, including technological, operation mode, and preprocessing [24]:

1. *Clustering*. The central assumption in this class of methods is that the message types coincide in similar groups. Various clustering methods with proper string-matching distances have been used. LKE applies weighted edit distance with hierarchical clustering to do log key extraction and a group splitting strategy to fine-tune the obtained log groups [9]. LogMine creates a hierarchy of log templates that allows the user to choose the description level of interest [14].

2. *Frequent pattern mining* assumes that a message type is a systematic set of tokens that appear throughout the logs. The procedures involve creating frequent sets, grouping the log messages, and extracting message types. Representative methods for this class are LFA [23], and LogCluster [34].

3. *Evolutionary*. This class member is MoLFI, which uses an evolutionary approach to find the Pareto optimal set of message templates [22].

4. *Log-structure heuristics*. These methods exploit different properties that emerge from the structure of the log. The state-of-the-art Drain assumes that the words do not vary too much at the beginning of the logs [16]. It uses this assumption to create a tree of fixed depth that can be easily modified for new groups. As a result, it produces the best results among the different adopted techniques [36].

5. *Longest-common sub-sequence* uses the longest common sub-sequence algorithm to dynamically extract log patterns from incoming log messages. Here, the most representative algorithm is Spell [7].

6. *Neural*. The key idea for this parsing is that the correct prediction of the masked word means that the word is a part of the log template; otherwise, it is a log parameter. The most representative approach is a self-supervised method NuLog [24], which utilizes the transformer architecture [6].

4 Event Time-Series Creation

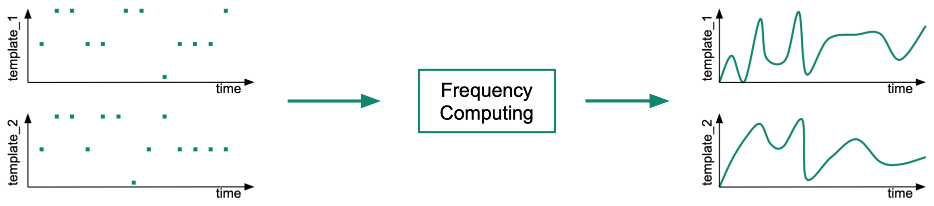


Fig. 3. The time-series creation flow.

After extracting the log templates, we construct a set of event time-series generated by each log template from all log data. In other words, we compute the frequency of log templates, that is, the number of appearances of one log template in each time bin (see also Figure 3). This transformation from log space to metric space helps us run causal inference algorithms directly on information from log data [18].

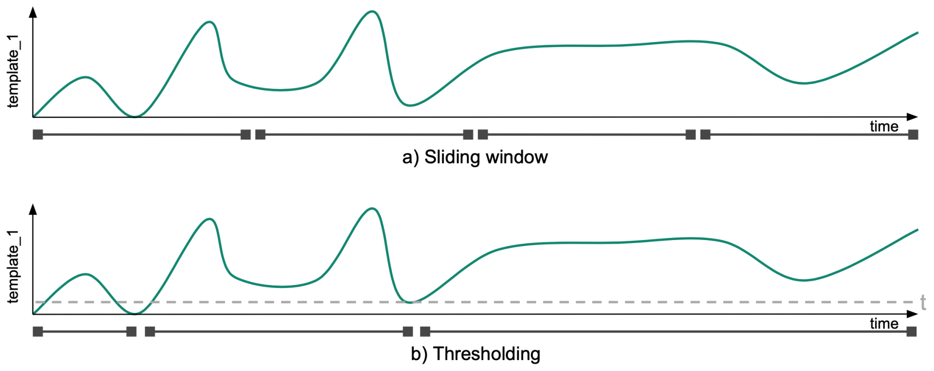


Fig. 4. The time-series modeling strategies.

There are two main strategies for treating time-series data [10]. One is to partition the data into non-overlapping equals windows and take the measurements in each unit as a data analytic unit (see also Figure 4 a). Another is to assume or estimate the number of lagged effects and treat all observations separated by no more than that number of log templates τ (threshold) as a data analysis unit (see also Figure 4 b). Each strategy has its disadvantages. The sliding window method necessarily omits relations across windows, and results may vary with the choice of window size. In the second method, the time units are not all independent, but most units are.

5 Causal Inference Algorithms

Causal inference consists of two key ideas to estimate causal relations between events: removing spurious correlations and determining directions of causal edges. This section explains the basic idea of causal inference and existing approaches to analyzing log data. I will illustrate three main classes of algorithms: constraint-based, score-based, and functional-based. In this paper, I selected one prominent algorithm from each group. But if we start exploring the multivariate time-series analysis literature regarding causality, we will likely come across Granger causality. This paper will also explain Granger causality and why it is not the true causality, unlike "cause and effect" relationships.

5.1 Constraint-based

The primary and main idea of a theory of inferred causation is to present a causal structure among variables in an acyclic-directed graph (DAG) called a causal graph in which arrows indicate causal orders, where each node represents a feature from the data set [27]. If we want to say that X causally influences Y and not vice versa, we write $X \rightarrow Y$.

But firstly, I will define the statistical decision procedure, a hypothesis test for **conditional independence**, to reveal causality from correlation. Conditional independence is usually formulated in terms of conditional probability. Assume three events: X, Y, and Z. X and Y are conditionally independent for given Z if

$$P(X \mid Y, Z) = P(X \mid Z) \tag{1}$$

where the events X and Y are independent as long as Z appears. In other words, since the probability of X given Z is the same as the probability of X given both Y and Z, this equality expresses that Y contributes nothing to the certainty of X. With conditional independence, we can remove pseudo-causality/spurious correlation in this way.

To clarify, neither conditional independence of two variables implies their independence nor vice versa. I prepared two counterexamples (see also Figure 5). The first example is about a child. Firstly, we focus only on the child's height and the number of words the child knows. When the first parameter is high, the second is high, too. So, they are not independent. But if we provide the child's age. We will see the dependence between age and height and between age and vocabulary. But if we go back to the probability definition, we can say that the height and vocabulary of the child are conditionally independent, given the child's age. The second example is about dice. We can throw dice twice and get two separate

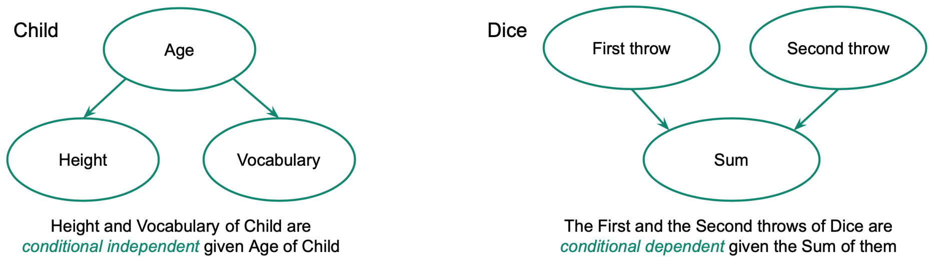

Child
Age
Height
Vocabulary
Height and Vocabulary of Child are *conditional independent* given Age of Child
Dice
First throw
Second throw
Sum
The First and the Second throws of Dice are *conditional dependent* given the Sum of them

Fig. 5. The conditional independence examples.

results. But if we provide the parameter like the sum of these throws, we will see the pairwise dependency between each throw and sum. And we can say in this case that the first and the second throw of dice are conditionally dependent on their sum.

One can consider three methods for testing conditional independence in practice [19]:

1. The *G-square test* is a method to evaluate conditional independence of binary or multi-level data based on information theory, using conditional cross-entropy (CE) and the data length (m). The G-square statistic G^2 is defined as:
$$G^2 = 2mCE(X, Y \mid Z) \tag{2}$$

2. The *Fisher-Z test* evaluates the conditional independence of continuous data based on Pearson's correlation coefficient (r) and the data length (m). The statistic Zs is defined as:
$$Zs = \frac{\sqrt{m - |Z| - 3}}{2} \log \frac{1+r}{1-r} \tag{3}$$

3. The *conditional mutual information test* evaluates conditional independence of discrete and continuous data based on information theory, using conditional entropy and probability function. The conditional MI test I is defined as:
$$I(X, Y \mid Z) = H(X \mid Z) - H(X \mid Y, Z) = E\left[\log \frac{p(X, Y \mid Z)}{p(X \mid Z)p(Y \mid Z)}\right] \tag{4}$$

In experiments with log data, Kobayashi et al. [19] found that Fisher-Z leaves more false-positive edges and takes more time than the G-square test.

This class of algorithms relies on four assumptions to develop the correct DAG structure. All the independencies in a graph need to be under the conditional independence criterion [25]:

1. *Causal Markov* – if two variables are conditional independent of each other, they must be unconnected given all their intermediate variables.
2. *Causal Faithfulness* – if two variables are conditionally dependent, connecting those variables must be an edge.
3. *Causal Sufficiency* – all the common causes of a pair of variables are measured (no hidden/external causes).
4. *Independency* – no variable is an (indirect) cause of itself.

Constraint-based methods for causal discovery have the advantage of being generally applicable. Still, the disadvantages are that faithfulness is a strong assumption and may require substantial sample sizes to get good conditional independence tests.

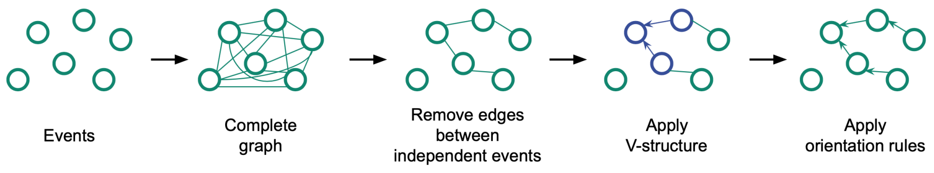


Fig. 6. The PC algorithm flow.

PC Algorithm [31]. One of the most common algorithms in literature is the PC algorithm, which aims to find the graphs representing precisely the independent relationships in the data through hypothesis tests. The PC algorithm consists of four steps (see also Figure 6). We start with a set of created from logs event time-series. Then, we construct a complete (fully connected undirected) graph from events. Next, we eliminate edges between conditionally independent nodes by analyzing all pairs. Then, the triple of nodes (in blue color on Figure 6) is called V-structure with the following orientations of edges. So, we look for the V-structures of nodes and determine the edge direction. Finally, we determine other edge directions with the orientation rules defined by the algorithm. However, some edges can be undirected even after applying the PC algorithm if one does not have enough information to determine edge directions. In other words, in the first two steps, we estimate a skeleton graph that is an undirected causality graph with the idea of causal inference. In the last two steps, we determine the directions of the detected edges with established rules. The V-structure is the main rule to decide on the direction of an edge. For example, three events X, Y, and Z are part of a graph $X - Y - Z$. X and Y are correlated, and Y

and Z are connected. One obtains a causal relationship $X \rightarrow Y \leftarrow Z$ if X and Z are not conditionally independent for Y. The remaining undirected edges can be oriented using experimental data or knowledge about data sources.

In practice, the PC algorithm has some variations. One of these algorithms is the PC-stable [5]. This algorithm solves the problem that the traditional PC depends on the order in which the events are given to the algorithm. This adaptation modifies how the algorithm creates the skeleton: in each level, the events that should be removed are saved in a queue and only permanently removed in the next iteration. PC-simple modification is based on local causality discovery [3]. The difference is that it only analyses the strongly related variables to the selected target variable (that is, it does not analyze or create the network with the remaining variables). This algorithm was developed to deal with high-dimensional data. The HITON-PC combines two previously presented algorithms, PC-stable and PC-simple [1]. Another extension is the parallel-PC, which is a parallel adaptation of PC [20]. This parallelization is done in the conditional independence tests performed at each algorithm level. These tests are grouped and distributed in different cores. The MPC modifies the original orientation rules by adding a new rule that prevents the creation of cycles (although DAGs do not allow cycles, PC does not explicitly prevent their creation) [33].

5.2 Score-based

These algorithms differ from the previous ones because they are based on the greedy strategy to make the locally optimal choice at each stage and compare the interim graphs through the Bayesian Information Criterion (BIC) [32]. These algorithms are usually costly in terms of performance since they have to score every model.

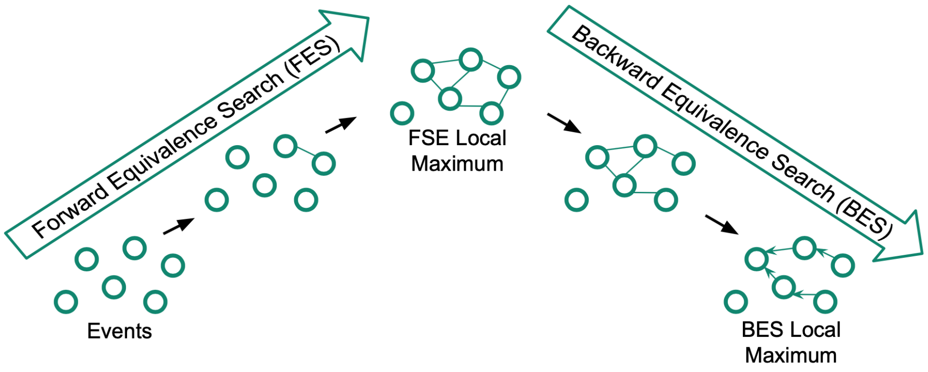

Forward Equivalence Search (FES)
FSE Local Maximum
Backward Equivalence Search (BES)
Events
BES Local Maximum

Fig. 7. The Greedy Equivalence Search flow.

Greedy Equivalence Search (GES) [4]. The GES algorithm generally consists of two phases (see also Figure 7). In the Forward Equivalence Search (FES) phase, the algorithm starts with a set of events. Then, at each step, as a decision is made whether adding an edge to the graph will increase the score BIC, the edge that most improves the score is added until the algorithm reaches a local maximum. This algorithm analyses all possible node-neighbors each time. Then, the second phase of the algorithm begins, which asks, edge by edge, which edge removal, if any, will improve the score until it reaches a local maximum. But in this phase, the direction of the edges is also considered.

In practice, the GES algorithm also has some variations. One example is the Greedy Interventional Equivalence Search (GIES), in which, besides the FES and BES phases, there is a third phase called the Turning Step [15]. In this extra phase, the algorithm iterates all possible orientations of edges without losing edges or creating new ones so that the previous graph can be reconstructed by only changing arrows. This phase was added with the intent to enhance estimation. Another example is the Fast Greedy Equivalence Search (FGS) [28]. This algorithm uses parallelization to optimize the original GES and applies a limited faithfulness assumption.

5.3 Functional-based

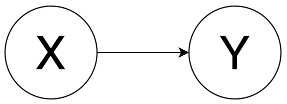


Fig. 8. The simple causal graph.

For the introduction to this class, I will start with defining the problem by determining the causal direction in the two-variable case, where no conditional independence relationship is available. A fundamental issue is given two variables X, Y (see also Figure 8), how to distinguish cause from effect. To do so, we must find a way to capture the asymmetry between them. Functional-based approaches represent the effect Y as a function of the direct causes X and some unmeasurable factors or noise:

$$Y = f(X) + E \tag{5}$$

where E is assumed to be independent of X, the function f explains how Y is generated from X. We assume that the transformation from (X, E) to (X, Y) is invertible, so E can be uniquely recovered from the observed variables X and Y. So, these algorithms consider the data asymmetry effect induced by the causal directions [10].

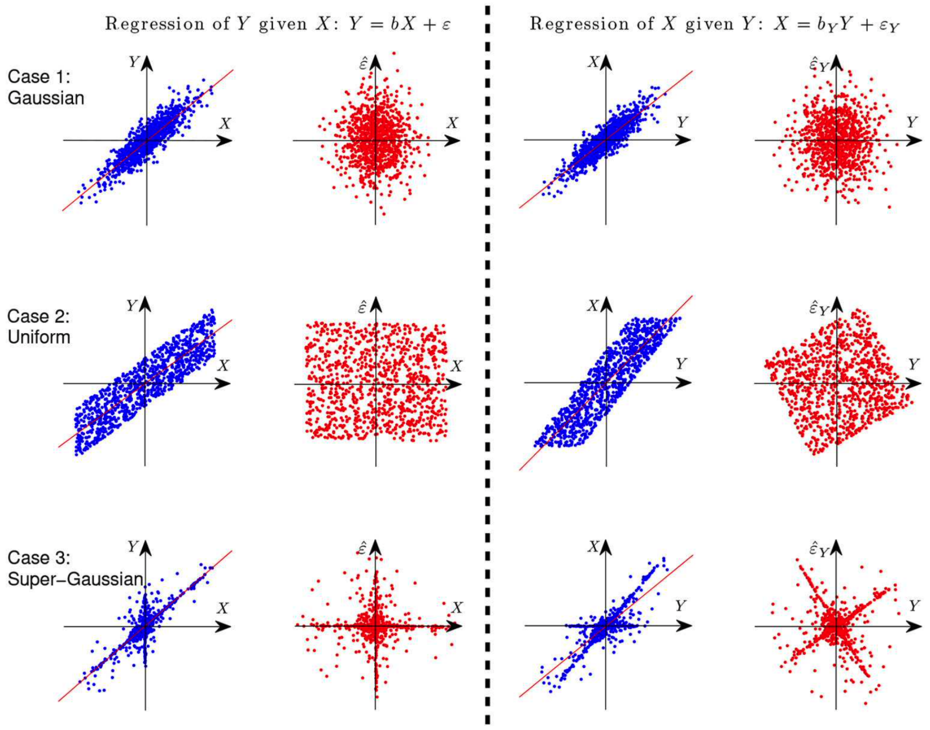


Fig. 9. The illustration of causal asymmetry between two variables with linear relations.

What does it mean by the data asymmetry effect? Under the above assumptions, we fit the functional-based test and then check for independence between the estimated noise term and the hypothetical cause. So, the direction that gives an independent noise term is considered plausible. The figure 9 is a simple example of why it is possible to identify the causal direction between two variables in the linear case, f is the linear function. Assume Y is generated from X in a linear form, where E is independent of X. Columns 1 and 3 are the scatter plots of the two variables X, and Y, the red line is the linear regression of Y on X, and columns 2 and 4 are the predictor and regression residual for two different regression tasks. The three rows correspond to different settings: X and E are both Gaussian (case 1), uniformly distributed (case 2), and distributed according to some super-Gaussian distribution (case 3). In the last two cases, X and E are non-Gaussian, and one can see clearly that for regression of X given Y (the backward direction), the regression residual is no longer independent of the predictor. However, they are uncorrelated by the construction of regression. In other words, in those two situations, the regression residual is independent of the predictor only for the correct causal direction, giving rise to the causal asymmetry between X and Y. As a result, based on this asymmetry, we can conclude that X is the cause of Y [10,11].

In practice, there are two directions of functional-based methods. The first way uses a linear function (e.g., Linear Non-Gaussian Acyclic Model (LiNGAM) [30]), and the second way — non-linear functions (e.g., Post-nonlinear (PNL) Causal Model [35]). In the linear, non-Gaussian, and acyclic model (LiNGAM), a function is linear, and at most, one of the noise terms E and cause X is Gaussian. In PNL, the effect Y is further generated by a post-nonlinear transformation on the non-linear effect of the cause X plus noise term E, where both functions are non-linear, and the outer function is assumed to be invertible.

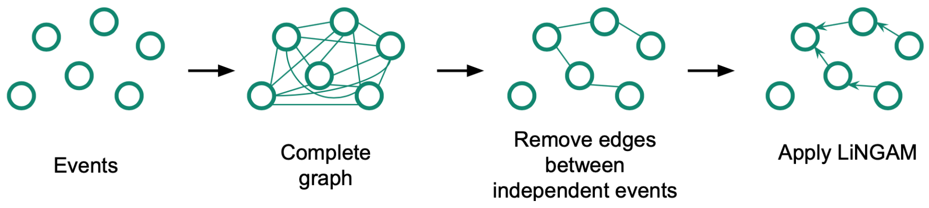


Fig. 10. The functional-based approach flow.

To apply this group of algorithms to our case, we first need to create the graph skeleton with a PC algorithm (see also Figure 10). Then, we estimate the orientation of edges based on the data asymmetry effect. Kobayashi et al. [17] found that MixedLiNGAM discovers ten times more causal edges than the PC algorithm.

5.4 The Granger Causality

The Granger causality test is a statistical hypothesis test for determining whether one time-series helps forecast another. Figure 11 is an example where the patterns in X are approximately repeated in Y after some time lag (two arrows). Thus, past values of X can be used to predict future values of Y. Clive Granger defined the causality relationship based on two principles: the cause occurs before the effect, and the cause has unique information about the future values of its effect [12].

To test the Granger causality, we need to augment the autoregression model of one time-series Y by including lagged values of another time-series X:

$$y_t = c + \epsilon_t + \sum_{i=1}^{p} \alpha_i y_{t-i} + \sum_{i=1}^{p} \beta_i x_{t-i} \tag{6}$$

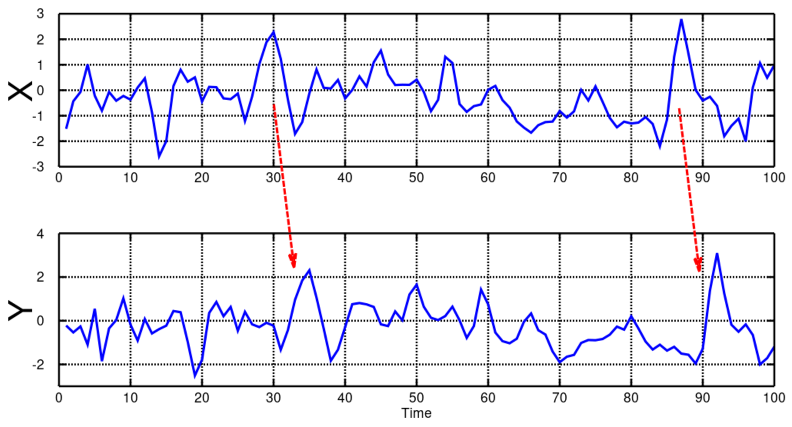


Fig. 11. The time-series X Granger-causes time-series Y.

So, we need to determine if any lags of X are statistically significant in our model of Y. We can do this using a Wald test F-statistic for linear restrictions. The Wald test is based on the relatively simple premise that we wish to compare the performance of a restricted model for Y, which excludes X, against an unrestricted model for Y, which includes X. The null hypothesis that X does not Granger-cause Y is accepted if and only if no lagged values of X are retained in the regression according to an F-statistic.

However, Granger stressed that some studies using "Granger causality" testing in areas outside economics reached "ridiculous" conclusions. *"Of course, many ridiculous papers appeared,"* he said in his Nobel lecture [13]. Thus, the Granger causality is not the true causality. It provides no insight into the relationship between the variables and is helpful only in the forecasting case. Moreover, time is the key parameter in this approach. However, the timestamp of system logs or time-series is not always reliable for determining causal directions due to time synchronization errors between devices, jitter, and network failures [19]. This kind of approach has other limitations. One is that the Granger causality only gives information about linear features. Besides this, one more is that the variables must be stationary. Finally, this approach depends on the observations we are dealing with [29].

6 Conclusion

Understanding causal relations helps construct interventions to achieve particular objectives and makes predictions under interventions. The growing accumulation of vast amounts of data necessitates the development of scalable automatic causal search algorithms. This paper has reviewed the step-by-step solution to the time-series data causal discovery challenge. Specifically, I first discussed the preparation phase for the set of log messages because causal discovery algorithms require a suitable structure as input. In this part, I discussed the log parsers and two strategies to create time-series data from log messages. Then, I described four classes of causal inference algorithms: constraint-based, score-based, functional-based methods, and the Granger causality test. If you have a task to explore causality, you can choose one of the proposed algorithms for each step. As a result, you get your algorithm pipeline. Future research directions include solving the missing relationships between two windows by time-series creation and integrating weights to directed edges to discuss the likelihood of causal impact [17].

References

1. Aliferis, C.F., Tsamardinos, I., Statnikov, A.: Hiton: a novel markov blanket algorithm for optimal variable selection. In: AMIA annual symposium proceedings. American Medical Informatics Association (2003)
2. Aussel, N., Petetin, Y., Chabridon, S.: Improving performances of log mining for anomaly prediction through nlp-based log parsing. In: 2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS) (2018). https://doi.org/10.1109/MASCOTS.2018.00031
3. Bühlmann, P., Kalisch, M., Maathuis, M.H.: Variable selection in high-dimensional linear models: partially faithful distributions and the pc-simple algorithm. Biometrika (2010). https://doi.org/10.1093/biomet/asq008
4. Chickering, D.M.: Optimal structure identification with greedy search. Journal of machine learning research (2002)
5. Colombo, D., Maathuis, M.H., et al.: Order-independent constraint-based causal structure learning. J. Mach. Learn. Res. (2014)
6. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018). https://doi.org/10.48550/arXiv.1810.04805
7. Du, M., Li, F.: Spell: Streaming parsing of system event logs. In: 2016 IEEE 16th International Conference on Data Mining (ICDM) (2016). https://doi.org/10.1109/ICDM.2016.0103
8. Du, M., Li, F., Zheng, G., Srikumar, V.: Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security (2017). https://doi.org/10.1145/3133956.3134015
9. Fu, Q., Lou, J.G., Wang, Y., Li, J.: Execution anomaly detection in distributed systems through unstructured log analysis. In: 2009 ninth IEEE international conference on data mining (2009). https://doi.org/10.1109/ICDM.2009.60
10. Glymour, C., Zhang, K., Spirtes, P.: Review of causal discovery methods based on graphical models. Frontiers in genetics (2019). https://doi.org/10.3389/fgene.2019.00524
11. Goudet, O., Kalainathan, D., Caillou, P., Guyon, I., Lopez-Paz, D., Sebag, M.: Learning functional causal models with generative neural networks. In: Explainable

and interpretable models in computer vision and machine learning. Springer (2018). https://doi.org/10.1007/978-3-319-98131-4_3

12. Granger, C.W.: Testing for causality: A personal viewpoint. Journal of Economic Dynamics and control (1980). https://doi.org/10.1016/0165-1889(80)90069-X
13. Granger, C.W.: Time series analysis, cointegration, and applications. American Economic Review (2004). https://doi.org/10.1257/0002828041464669
14. Hamooni, H., Debnath, B., Xu, J., Zhang, H., Jiang, G., Mueen, A.: Logmine: Fast pattern recognition for log analytics. In: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (2016). https://doi.org/10.1145/2983323.2983358
15. Hauser, A., Bühlmann, P.: Characterization and greedy learning of interventional markov equivalence classes of directed acyclic graphs. The Journal of Machine Learning Research (2012)
16. He, P., Zhu, J., Zheng, Z., Lyu, M.R.: Drain: An online log parsing approach with fixed depth tree. In: 2017 IEEE international conference on web services (ICWS) (2017). https://doi.org/10.1109/ICWS.2017.13
17. Jarry, R., Kobayashi, S., Fukuda, K.: A quantitative causal analysis for network log data. In: 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC) (2021). https://doi.org/10.1109/COMPSAC51774.2021.00213
18. Jia, T., Chen, P., Yang, L., Li, Y., Meng, F., Xu, J.: An approach for anomaly diagnosis based on hybrid graph model with logs for distributed services. In: 2017 IEEE international conference on web services (ICWS) (2017). https://doi.org/10.1109/ICWS.2017.12
19. Kobayashi, S., Otomo, K., Fukuda, K., Esaki, H.: Mining causality of network events in log data. IEEE Transactions on Network and Service Management (2017). https://doi.org/10.1109/TNSM.2017.2778096
20. Le, T.D., Hoang, T., Li, J., Liu, L., Liu, H., Hu, S.: A fast pc algorithm for high dimensional causal discovery with multi-core pcs. IEEE/ACM transactions on computational biology and bioinformatics (2016). https://doi.org/10.1109/TCBB.2016.2591526
21. Lou, J.G., Fu, Q., Wang, Y., Li, J.: Mining dependency in distributed systems through unstructured logs analysis. ACM SIGOPS Operating Systems Review (2010). https://doi.org/10.1145/1740390.1740411
22. Messaoudi, S., Panichella, A., Bianculli, D., Briand, L., Sasnauskas, R.: A search-based approach for accurate identification of log message formats. In: 2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC) (2018)
23. Nandi, A., Mandal, A., Atreja, S., Dasgupta, G.B., Bhattacharya, S.: Anomaly detection using program control flow graph mining from execution logs. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2016). https://doi.org/10.1145/2939672.2939712
24. Nedelkoski, S., Bogatinovski, J., Acker, A., Cardoso, J., Kao, O.: Self-supervised log parsing. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases (2020). https://doi.org/10.1007/978-3-030-67667-4_8
25. Nogueira, A.R., Gama, J., Ferreira, C.A.: Causal discovery in machine learning: Theories and applications. Journal of Dynamics & Games (2021). https://doi.org/10.3934/jdg.2021008
26. Otomo, K., Kobayashi, S., Fukuda, K., Esaki, H.: Latent variable based anomaly detection in network system logs. IEICE TRANSACTIONS on Information and Systems (2019). https://doi.org/10.1587/transinf.2018OFP0007

27. Pearl, J.: Causality. Cambridge university press (2009)
28. Ramsey, J., Glymour, M., Sanchez-Romero, R., Glymour, C.: A million variables and more: the fast greedy equivalence search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images. International journal of data science and analytics (2017). https://doi.org/10.1007/s41060-016-0032-z
29. Seth, A.: Granger causality. Scholarpedia (2007)
30. Shimizu, S., Hoyer, P.O., Hyvärinen, A., Kerminen, A., Jordan, M.: A linear non-gaussian acyclic model for causal discovery. Journal of Machine Learning Research (2006)
31. Spirtes, P., Glymour, C.N., Scheines, R., Heckerman, D.: Causation, prediction, and search. MIT press (2000). https://doi.org/10.1080/08839514.2018.1526760
32. Stoica, P., Selen, Y.: Model-order selection: a review of information criterion rules. IEEE Signal Processing Magazine (2004). https://doi.org/Model-order selection: a review of information criterion rules
33. Tsagris, M.: Bayesian network learning with the pc algorithm: an improved and correct variation. Applied Artificial Intelligence (2019). https://doi.org/10.1080/08839514.2018.1526760
34. Xu, W., Huang, L., Fox, A., Patterson, D., Jordan, M.I.: Detecting large-scale system problems by mining console logs. In: Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles (2009). https://doi.org/10.1145/1629575.1629587
35. Zhang, K., Hyvarinen, A.: On the identifiability of the post-nonlinear causal model. arXiv preprint arXiv:1205.2599 (2012). https://doi.org/10.48550/arXiv.1205.2599
36. Zhu, J., He, S., Liu, J., He, P., Xie, Q., Zheng, Z., Lyu, M.R.: Tools and benchmarks for automated log parsing. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) (2019). https://doi.org/10.1109/ICSE-SEIP.2019.00021