Log File Anomaly Detection Based on Process Mining Graphs

Sabrina Luftensteiner(✉) and Patrick Praher

Software Competence Center Hagenberg, Hagenberg im Muehlkreis, Austria
sabrina.luftensteiner@scch.at
http://www.scch.at

Abstract. In process industry, it is quite common that manufacturing machines repeat a certain order of process steps consistently. These process steps are often declared in programs and have a linear workflow. During the production process, logs are generated for monitoring and further analysis, where not only production steps but also incidents and other deviations are logged. As the manual monitoring of such processes is quite time consuming and tedious, the demand for automatic anomaly and deviation detection is rising. A potential approach is the usage of Process Mining, whereat not all requirements are met. In this paper, we propose a new approach based on spectral gap analysis for the detection of anomalies in log files using the adjacency matrix generated by Process Mining techniques. Furthermore, the experiments section covers their application on a linear process and non-linear processes with deviating paths.

Keywords: Log files · Anomaly detection · Process mining

1 Introduction

Log files are created by devices or systems to provide information about processes and their behavior in a textual or granular view [4]. They do not only provide expected behavior, but also important information about possible incidents [7], e.g. via error logs, which may reveal system or process weaknesses [4]. The manual inspection of log files is often very time consuming and tedious, which demands for an automated approach [7]. Process Mining [14] aims at an automatic extraction of process knowledge, whereat event-logs are utilized to grasp the complex nature of industrial processes [5]. Especially in the field of manufacturing and programming of machines, the usage of Process Mining is not highly present although the area of application fits outstandingly well [13].

In manufacturing, it is common to program machines to repeat certain program workflows consistently. These programs usually have a linear workflow, which consists of a specific order of non-repeating steps. During the execution the

The research reported in this paper has been supported by the Austrian Ministry for Transport, Innovation and Technology, the Federal Ministry for Digital and Economic Affairs, and the Province of Upper Austria in the frame of the COMET center SCCH.

steps and incidents are logged constantly. These logs are evaluated using Process Mining to determine visually, if a process workflow is as expected using certain techniques for graph generation and conformance checking. As the monitoring may be very time consuming and tedious, new approaches for the comparison of such Process Mining graphs are required. It is in demand to provide computationally efficient, yet accurate comparison possibilities, to match workflows and their deviations. In the course of this paper, linear processes represent optimal processes, where anomalies have to be detected. Slightly non-linear process executions may still reflect a good execution, e.g. due to error messages with no direct influence on the execution or process, therefore a comparison to linear workflows may not be sufficient enough. The detection of strong anomalies or deviations, which is reflected by a changing process execution behavior, is significant as they may indicate wearing of machine parts or wrong usage/implementation of programs as well as adapted programs.

This paper proposes an approach to detect anomalies in linear processes using Process Mining techniques and event-logs generated using log files. The approach is based on the spectral gap calculation and uses the adjacency matrix created by Process Mining techniques. Furthermore, an insight into related work is provided, presenting the current state of the art as well as disadvantages of existing approaches. The experimental section covers the effectiveness of the approach in differing scenarios, whereat different deviation types are tackled.

1.1 Discussion on Existing Approaches and Related Work

This subsection provides insight into related work for the comparison of event-logs to find behavior changes or anomalies in linear processes. Various approaches for anomaly detection in log files, conformance checking and process deviance are presented, highlighting their drawbacks regarding our use-cases.

Anomaly Detection in Logs. Frei et al. propose a log file visualization called Histogram Matrix (HMAT), which visualizes the content of textual log files to enable the detection of anomalies using a combination of graphical and statistical techniques [7]. An approach for the detection of abnormal behavior in event-logs of social network websites is provided by Sahlabadi et al. [12]. In a first step the normal user behavior pattern is defined using a genetic Process Mining technique, the second step covers the detection of abnormal behavior via a fitness function, which is time as well as resource consuming. Breier et al. base their anomaly detection approach on data mining techniques for log analysis and the creation of dynamic rules [4].

The presented methods are quite sensitive to deviating process executions and are likely to not work with frequently deviating executions with no stable base, e.g. due to error messages.

Conformance Checking. Conformance checking is used to detect inconsistencies between a process model and its corresponding execution log [6]. Various approaches for conformance checking are available, one of which is provided by Dunzer et al. [6] covering modelling languages, algorithmic types, quality metrics as well as perspectives. An incremental approach is proposed by Rozinat et al. [11] to check the conformance of a process model and an event log by measuring their fitness and analyzing the appropriateness from a structural as well as behavioral perspective. Adriansyah et al. [1] present a robust method for conformance checking using various metrics, which do not cover unsatisfied or unhandled events. Furthermore, a robust replay analysis technique for measuring the conformance is provided Adriansyah et al. [2], which also provides intuitive diagnostics such as skipped or inserted activities.

The conformance checking approaches seem quite interesting, but they require a stable process model as a base to provide valid results. In manufacturing, this may often be not the case as processes are adapted or unpredictable errors are logged, which complicates the automatic creation of a good process model without human intervention.

Process Deviance and Variants. Process deviance and variants provide a third possibility to inspect process workflows and their conformance. The usage of process model variants is rising due to the increased monitoring of dynamic processes. As they are rather difficult to maintain, various methods for generalizing and the deviance from this generalization have been proposed by researchers. Li et al. proposed papers dealing with the goals [9] and issues [8] related to the mining of process model variants with a focus on the integration of process changes into generic process models. Another approach by Li et al. [10] covers the learning from process model adaptations and the discovery of reference models, which are used to configure variants with minimum effort. Process variability and the handling of variants in such varying processes was dealt with by Bolt et al. [3]. They developed an unsupervised technique to detect significant variants in event-logs to detect if variants represent the same process or differ.

Process deviance and variants provide suitable approaches to handling variants and the generation of a reference model. Still, human intervention is likely needed to generate a meaningful base for further usage and to define normal behavior of a process. Furthermore, human input is needed to define positive and negative deviations of the process.

2 Methods

This section describes the preparation of event-logs and our method used for the detection of changing behavior. The subsection covering the data preparation will explain the transformation of event-logs to adjacency matrices, which are used as basis for the spectral gap approach.

2.1 Preparation of Event-Logs

Table 1 contains a simplified event-log of a process. The relevant information covers CaseID, Activity and Timestamp. The CaseID resembles the unique identifier of one process execution, the activity covers a textual representation of a process step and the timestamp is used to keep the data sorted. The data is extracted from log files, which were gathered in industrial environments, e.g. machines or systems. In our use-cases, the events do not include flags to indicate the state of an activity, e.g. start or completed, but rather independent activities, which have to be interpreted correctly. CaseIDs may have overlapping timestamps as machines may already start a new production cycle while the other one is still running.

Table 1. Simplified event-log including unique identifier, activity and timestamp.

CaseID	Activity	Timestamp
1	T1 - Insert workpiece	2022-04-01 00:00:00
1	T2 - Apply heat	2022-04-01 00:00:10
1	T3 - Bend	2022-04-01 00:00:20
1	T2 - Apply heat	2022-04-01 00:00:30
1	T3 - Bend	2022-04-01 00:00:40
1	T2 - Apply heat	2022-04-01 00:00:50
1	T2 - Apply heat	2022-04-01 00:01:00
1	T3 - Bend	2022-04-01 00:01:10
2	T1 - Insert workpiece	2022-04-01 00:01:20

The first step of the data preparation is the extraction of an adjacency matrix, which is extracted from the event-logs using Process Mining techniques to create a directed graph. The values within the adjacency matrix have to be relative in our use-case, referring to the relative amount of edge transitions between nodes. Absolute values complicate further comparisons and evaluations as the processes would have to consist of an identical amount of executions for each comparison timespan. Table 2 contains the matching relative adjacency matrix of the event-log presented in Table 1.

2.2 Spectral Gap Calculation Using Event-Logs

Prior to calculating the spectral gap, the adjacency matrix has to be made doubly stochastic. These matrices fulfill the requirements of being a square matrix

Table 2. Adjacency matrix.

	T1	T2	T3
T1	0	1	0
T2	0	0.25	0.75
T3	0.33	0.66	0

with non-negative real values, having row and column sums of 1, having matrix elements between 0 and 1 as well as the largest eigenvalue being always 1. Equation 1 and Eq. 2 cover the steps to create a doubly stochastic matrix (DSM) from a square matrix M:

$$SM = M + M^T \tag{1}$$

$$DSM_{i,} = \frac{SM_{i,}}{\sum SM_{i,}} \tag{2}$$

At first a transposed matrix is added to the original matrix to generate a symmetric matrix (SM). The symmetric matrix is then normalised and produces a doubly stochastic matrix (DSM), which is used to calculate eigenvalues. The eigenvalues result in the subtraction of the second largest eigenvalue from the largest eigenvalue to receive the spectral gap, see Eq. 3 where λ_1 refers to the largest and λ_2 to the second largest eigenvalue of DSM. The process of calculating the spectral gap can be summarized as follows:

1. Transformation of event-logs to relative adjacency matrices
2. Transformation of adjacency matrices to doubly stochastic matrices
3. Calculation of eigenvalues for each matrix
4. Subtraction of second largest from largest eigenvalue to calculate spectral gap for each matrix
5. Comparison of spectral gaps

$$SG = \lambda_1 - \lambda_2 \tag{3}$$

The calculation of the spectral gap does not include further knowledge of the basic structure or additional information of prior process workflows. It only considers what is available at that point in time and how linear the workflow is. The more deviations from the originally linear process path, the higher the spectral gap. Using this approach on usually linear workflows, it is more straightforward to detect anomalies or deviations in the process flow.

3 Experiments and Discussion

This section covers the experimental setup as well as different types of process graph changes and their influence on our spectral gap approach. The overall goal is to demonstrate that using these methods on event-logs, we are able to detect behavior changes or anomalies in linear process executions.

3.1 Event-Logs Structure

A program consists of a sequence of routines or process steps, whereat the execution is logged in log files for the creation of event-logs. Steps within a program refer to activities of a machine to manufacture a product, ranging from laser

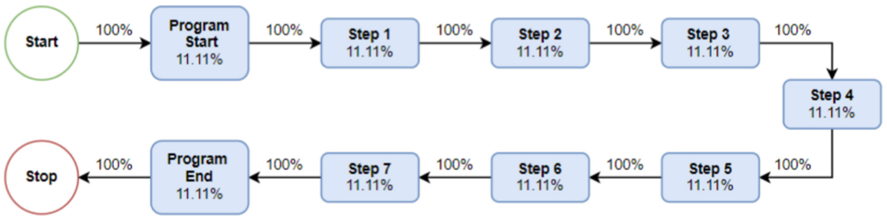

Fig. 1. Workflow example of a flawless production cycle with no repeating activities or additional suspicious activities, e.g. error messages.

cutting to coating to bending and various other activities. Additional errors or warnings may be logged during the process, e.g. occurring due to faulty executions, wrong program usage or wearing of machine parts. The stored event-logs are analyzed on a daily basis, using the execution start of a program as ID for each process cycle. The following event-log activities are covered in our scenario:

– Start Program - Start of a new program cycle
– Stop Program - Stop of a program cycle
– Step x - Process step within a program
– Error x - Error appearing during a program execution

Table 3 covers the simplified event-log of one program execution.

Table 3. Simplified event-log of one program execution.

CaseID	Activity	Timestamp
1	Start program	2021-07-01 00:00:00
1	Step 1	2021-07-01 00:00:10
1	Step 2	2021-07-01 00:00:20
1	Step 3	2021-07-01 00:00:30
1	Step 4	2021-07-01 00:00:40
1	Step 5	2021-07-01 00:00:50
1	Step 6	2021-07-01 00:01:00
1	Step 7	2021-07-01 00:01:10
1	Stop program	2021-07-01 00:01:20

Figure 1 provides a visualization of a manufacturing process using Process Mining on event-logs, whereas Table 3 contains the basic event-log. The event-log contains an optimal and repeating workflow, which covers a start event, a stop event and seven intermediate process steps for each execution. As no deviations are available, the graph represents a linear process without any branching and is therefore used as the basis for further experiments with deviating workflows.

The following list covers deviation types in the program execution, which occur in industrial environments:

– *Repeating Activities* occur due to unsatisfactory outcomes from one step execution or the reduction of program code, e.g. if one step has to be executed twice.
– *Additional Activities* appear either planned, e.g. due to the adaption of existing programs, or unplanned, e.g. due to error or warning messages.
– *Combination of Additional & Repeating Activities* indicate problems within the program or machine if the combination appears unplanned.
– *Program Interruptions* occur often in combination with (unplanned) additional activities, such as errors and lead to an early termination of the execution process.

3.2 Experimental Setup and Evaluation

The experimental scenario consists of a linear program, which was artificially created and covers 12 executions a day for a specific amount of days. A normal process execution has a linear workflow consisting of 7 steps and a likelihood of 10% for an error message to appear. Those additional error messages reflect a typical behavior in production as it is very likely that some incidents or disturbances are happening during a process execution, e.g. wrong alignment of material. A slightly non-linear workflow may still reflect a good execution, e.g. due to error messages with no direct influence, therefore a comparison to linear workflows may not be sufficient enough. The timespan of our scenario covers 7 months to simulate a more realistic area of application. Anomalous days have different characteristics regarding occurring anomalies, with some days having more anomalous behavior and others less, e.g. the occurrence of additional errors.

The scenario covers a timespan of 7 months containing anomalous and non-anomalous data. The individual days cover process executions with weaker and stronger deviations, which indicate problems with either the machine or program as the usual behavior is negatively influenced and changed. The anomalous days may vary regarding their anomalous activity, some having more and others having less different behavior. The sample adjacency matrix is generated for each day and our spectral gap approach is applied for each these matrices. Figure 2 visualizes the results using our approach, whereat the days are colored according to their dedicated category. The results are presented as standard deviations. The visualization shows that it is possible to detect the anomalous days due to their differing behavior using the spectral gap method and a fitting threshold, which is set to the mean and $2 * \sigma$ of the first 20 days. The anomalous days are clearly visible as they have higher sigmas due to their strong deviations in comparison to normal behaving executions.

3.3 Discussion

The strength of the spectral gap approach is its uncomplicated and fast computation. In contrast to other approaches, it is not necessary to know all process

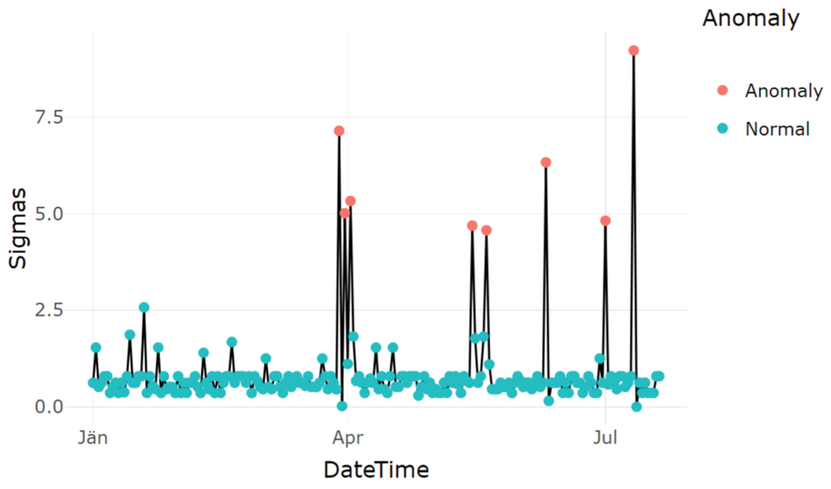

Fig. 2. Visualization of results regarding a repeating process over 8 months, using the standard deviation σ as indicator.

parts in advance and therefore the used adjacency matrices may have different dimensions. Our approach is able to indicate if programs have a linear execution workflow, meaning the more deviating branches are available in the workflow the higher the spectral gap will be. In case of our linear program example, the spectral gap approach enables us to evaluate if the process operates well or if problems are occurring and the program workflow is deviating from its original course. Another advantage of the spectral gap approach is that no common base has to be calculated prior to the actual calculation, increasing the computational efficiency and reducing the overall computation time.

On the downside, the usage of the spectral gap approach is not suitable for non-linear programs, which contain e.g. cycles or elemental deviations within the optimal process path. As only one matrix is focused on, no relations or similarities of multiple matrices, e.g. spanning over a few days, are extracted. Considering this use-case, deviations in processes may not be detected, e.g. if a program is changed at the beginning of a day.

4 Discussion and Conclusion

In our work, we addressed the problem of detecting anomalies and workflow deviations within linear programs. At first, a description of existing approaches and state of the art was presented, highlighting their purpose and disadvantages regarding our use-case. Building up on Process Mining techniques and event-logs, a spectral gap approach was developed to detect anomalies within process executions of industrial machines based on a linear workflow. The experiments section provided a specific setup consisting of a basic linear program and its possible deviations to demonstrate the impact on the methods and their effectiveness. Our method allows the unsupervised identification of flawless and anomalous linear machine executions without the need for any hyper parameter optimization.

As our method only works in combination with linear combinations, non-linear approaches should also be addressed in future work. Further possibilities are experiments with other anomaly detection methods in combination with adjacency matrices as well as the generation and identification of a training phase for supervised approaches.

References

1. Adriansyah, A., van Dongen, B.F., van der Aalst, W.M.P.: Towards robust conformance checking. In: zur Muehlen, M., Su, J. (eds.) BPM 2010. LNBIP, vol. 66, pp. 122–133. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20511-8_11
2. Adriansyah, A., van Dongen, B.F., van der Aalst, W.M.: Conformance checking using cost-based fitness analysis. In: 2011 IEEE 15th International Enterprise Distributed Object Computing Conference, pp. 55–64. IEEE (2011)
3. Bolt, A., van der Aalst, W.M.P., de Leoni, M.: Finding process variants in event logs. In: Panetto, H., et al. (eds.) OTM 2017. LNCS, vol. 10573, pp. 45–52. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69462-7_4
4. Breier, J., Branišová, J.: Anomaly detection from log files using data mining techniques. In: Kim, K.J. (ed.) Information Science and Applications. LNEE, vol. 339, pp. 449–457. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46578-3_53
5. Corallo, A., Lazoi, M., Striani, F.: Process mining and industrial applications: a systematic literature review. Knowl. Process. Manag. **27**(3), 225–233 (2020)
6. Dunzer, S., Stierle, M., Matzner, M., Baier, S.: Conformance checking: a state-of-the-art literature review. In: Proceedings of the 11th International Conference on Subject-Oriented Business Process Management, pp. 1–10 (2019)
7. Frei, A., Rennhard, M.: Histogram matrix: log file visualization for anomaly detection. In: 2008 Third International Conference on Availability, Reliability and Security, pp. 610–617. IEEE (2008)
8. Li, C., Reichert, M., Wombacher, A.: Issues in process variants mining (2008)
9. Li, C., Reichert, M., Wombacher, A.: Mining process variants: goals and issues. In: 2008 IEEE International Conference on Services Computing, vol. 2, pp. 573–576. IEEE (2008)
10. Li, C., Reichert, M., Wombacher, A.: Mining business process variants: challenges, scenarios, algorithms. Data Knowl. Eng. **70**(5), 409–434 (2011)
11. Rozinat, A., Van der Aalst, W.M.: Conformance checking of processes based on monitoring real behavior. Inf. Syst. **33**(1), 64–95 (2008)
12. Sahlabadi, M., Muniyandi, R.C., Shukur, Z.: Detecting abnormal behavior in social network websites by using a process mining technique. J. Comput. Sci. **10**(3), 393 (2014)
13. Son, S., et al.: Process mining for manufacturing process analysis: a case study. In: Proceeding of 2nd Asia Pacific Conference on Business Process Management, Brisbane, Australia (2014)
14. Van Der Aalst, W.: Process mining: overview and opportunities. ACM Trans. Manage. Inf. Syst. (TMIS) **3**(2), 1–17 (2012)