Abstract

Process mining of unstructured and noisy environments for which there are no prescribed process models remains an important area of security and process mining research. This paper discusses an unsupervised, threshold-based method of anomaly detection and process mining using the iterative SUBDUE graph compression method and the Inductive Miner. We provide an overview of process mining definitions and existing approaches, then evaluate the method on synthetic data provided by a data generation algorithm. The method generates a dendrogram, a taxonomical representation of compressing process features by which further analysis can be performed. The detection of anomalies is performed within the local context of these features. This method provides a framework by which many unstructured and noisy organizational, technological, and natural processes may be monitored and modeled.

Introduction

As described in [1], a Process Aware Information System is, “…a software system that manages and executes operational processes involving people, applications, and/or information sources on the basis of process models” (p. 5). This formal definition identifies operational management systems as systems which are both aware of process data and likewise prescribe tasks and activities per prescribed process models. Thus, such systems incorporate process data and process models in a feedback loop whereby processes can be tracked, defined, and organized via process models; likewise, process models may be derived and analyzed based on process data.

In many real-world scenarios, however, institutions rely on overlapping but non-interoperable systems, enterprise management platforms, and tribal knowledge to monitor and control processes, such that the overarching requirements of formal PAIS’s become infeasible or intrusive. This paper focuses on contexts for which a PAIS is instead an abstraction consuming noisy process data derived from multiple systems, and from which process models are then mined and analyzed to improve processes. This is amenable to scenarios in which processes execute in an embedded and often non-stationary framework of changing people, tools, resources, and institutional knowledge, in the absence of prescribed process models. In this scenario, a PAIS is a collection of operational systems and disparate data sources by which one derives traces characterizing the underlying process-oriented view of processes and their activities.

The ability to mine and analyze normative process patterns in such unstructured contexts is critical for extracting actionable information and detecting unusual activity. Detection of anomalous activity requires prior normative activity patterns, so detection and normative pattern mining are complementary tasks. For this, we present a method for mining process patterns from workflow logs also exhibiting useful anomaly detection properties. We use the Inductive Miner [2] to construct a graphical process model from log data, then apply the SUBDUE [3] graph compression method to iteratively extract a hierarchical dendrogram of normative patterns of this model. In subsequent post-processing, anomalies and other useful features are discovered.

This hybrid approach is useful since the Inductive Miner extracts generality from process log data, outputting a graphical model *M* capable of generating all traces in a process log, regardless of noise in the input log. As such general models are typically overly inclusive, SUBDUE then constructs a hierarchy of structures of *M* most relevant to the log as a dendrogram, extracting from M only its most informative components. In short, for a given trace log, the Inductive Miner is used to mine a general process model, SUBDUE is then used to extract only the most specific components of the model. Using this unsupervised method, one can mine normative process patterns, detect anomalies to those patterns, and perform other analyses.

This captures the spirit and contribution of our method, but requires some definitions. From the control-flow perspective, common process mining terms can be framed in a familiar graphical manner:

* process model: A graph with vertices representing activities, and edges representing one-step transitions between activities. Processes can contain many constructs representing linear and non-linear constructs, and a variety of notations and languages have been defined over the space of process models. The popular example is the Petri-Net [4] [5].
* process grammar: Constructs defining common process traversal patterns. For example, an AND-SPLIT is a set of edges branching from a single node and traversing activities in parallel before synchronizing at some later activity. Other basic constructs include OR-SPLIT, XOR-SPLIT, LOOP, and JOIN [14] [15].
* workflow trace: A single execution of a process as a partially-ordered sequence of activities, following any valid path from a START node to an END node on a process model. These are represented as a string composed of letters representing the activities traversed.
* workflow log: A set of workflow traces, for which various measures are taken to mitigate properties of noise and incompleteness.
* process miner: Any algorithm for constructing a process model from a workflow log, per criteria such as specificity and generality. Specificity favors restrictive models including only or even less than the behavior described by the workflow log, whereas generality favors larger models describing all traces and potentially additional behavior.
* partial-order property [18]: A property of workflow traces where activities may be randomly-ordered with respect to parallel activities. ‘ABCD’ and ‘ACBD’ might be workflow traces from some model, where ‘C’ and ‘B’ are parallel sub-processes, and may recursively embody further parallel sub-processes, and ‘A’ always occurs before ‘D’. The primary task of process mining algorithms is disambiguating the partial-orderings, by applying rules and heuristics to generate models with desired properties of complexity, specificity, and generality. The enormous space of possible graphical models defined over a set of partially-ordered traces is a primary challenge for these algorithms.
* spaghetti model: A workflow defined by highly diverse, informal, and possibly disorderly behavior, typically containing many scattered, repetitive events. These represent unstructured business processes, in contrast to “lasagna” processes with prescriptive and stratified behavioral logic.
* inductive miner: The process mining algorithm that generates a graphical process model capturing the most general, all-inclusive view of the traces in some log. For our purposes, this model is used in pre-processing to convert a workflow log of traces into a collection of subgraphs [2].
* SUBDUE: Short for “Substructure Discovery”, this method implements a subgraph beam search over a graph collection and, by applying the minimum-description length (mdl) heuristic, returns the top-k most compressing sub-graphs [3].
* GBAD: Acronym for “graph-based anomaly detection” [6], this method internally calls SUBDUE, then implements methods for detecting anomalies occurring in the context of discovered patterns.

An in-depth overview of process mining terms and methods can be found in [1] and [7].

Previous Work

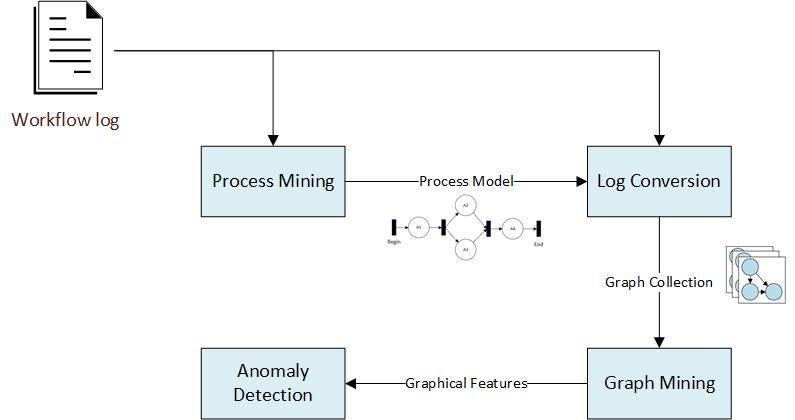
The SUBDUE graph-compression method accommodates any context of mining graph data for normative patterns, and hence works as a graphical feature detector. It was previously used for knowledge representation systems [1], and more recently in security applications for intrusion detection [8]. Using SUBDUE as a process mining tool has been successfully performed by [9] and more recently by Genga [10], whose results demonstrated the method’s utility for “spaghetti processes” describing more realistic institutional processes.

GBAD formalizes SUBDUE’s anomaly-detection capabilities, particularly within the immediate proximity of normative graphical patterns. This is appropriate for safety-critical and security contexts for which there exists some underlying process model by which normative patterns can be assumed to have a ground-truth policy, but less so when the underlying process is not prescribed. An example application is given by Holder and Eberle, in which GBAD was used for insider threat detection based on three separate anomaly detection algorithms [16].

Process-mining anomaly detection focuses primarily on the mining process itself and on trace-scoring schemes. W. van der Aalst [12] details scoring schemes, by which work traces are replayed on a discovered model, assigned a numeric fitness score, and the anomalies determined based on a discriminative threshold. Bezerra’s work examined anomaly detection using several threshold-based approaches within the process mining algorithm itself [11]. Bezerra decomposed this family of process-based anomaly detection into three groups: threshold-based, iterative, and sampling. Our approach does not fit squarely into one these categories since it is compression based: a generic process model is mined, graphical features detected, and anomalies are detected and reported in post-processing. Our work replicates Bezerra’s data generation methods, but otherwise adds to this work as just described.

The Method

Under our relaxed definition, a PAIS is a composition of process monitoring systems by which workflow traces are extracted for process mining in a standard log format, such as Extensible Event Stream (XES) [17]. Our method decomposes to three tasks: converting such a log to a collection of subgraphs via a mined process model, extracting descriptive normative graphical patterns of the log (with respect to this model) as a dendrogram, and lastly detecting outliers and anomalous behavior.



For the first task, the Inductive Miner was suitable for mining the most general graphical process model described by some log. This model is usually overly-inclusive, hence the second and third tasks discover the patterns and features precisely relevant to describe the log. For this task, we use the SUBDUE graph-compression method to discover normative behavioral patterns, subsequently permitting anomaly detection.

The workflow of these tasks extends to contexts in which one desires to discover the overall characteristics of a process without respect to prior constraints, such as prescribed process models. Our approach extends to more realistic and informal “spaghetti” model scenarios in which processes are organically-defined and highly unstructured: enterprises, communication networks, fraudulent or criminal networks, and so on.

Using Graph Compression to Discover Patterns and Cluster Traces

SUBDUE discovers highly compressing patterns in graph data via the minimum description length (mdl) principle and a beam search over candidate subgraphs. This property satisfies the requirement for an unsupervised method of discovering the most meaningful components of a graphical process model, since a workflow log is also a set of subgraphs generated by a process model. The Inductive Miner complements our approach by providing the super-graph for converting a log of partially-ordered traces into subgraphs; these subgraphs are passed to SUBDUE to discover meaningful graphical patterns.

Prior work on SUBDUE showed great potential when running the method iteratively on a set of graphs [8]. At each iteration, the most compressing subgraph discovered by SUBDUE was used to replace all such instances with a single node, and the method repeated until no further compression was possible. At the end, the authors obtained a recursive and hierarchical description of a set of graphs, by which they modelled their anomaly-detection activities.

We tested a similar approach using GBAD, by which workflow traces were iteratively recompressed using the most-compressing subgraph found at each iteration. The three anomaly detection methods of GBAD were used to detect anomalies at each iteration. While successful in terms of discovering patterns, this method suffered a high false positive rate for anomaly detection. Ultimately the issue was with iterative recompression: on successive iterations, the most highly compressing subgraph was often only a small alteration of node substitution, deletion, or insertion to a compressing subgraph found by a previous iteration. GBAD’s primary deficiency in this context is that its anomaly-detection methods apply to the local vicinity of the compressing pattern discovered by SUBDUE. Hence, the search space was highly redundant, often analyzing the same regions of the graph space, but failing to reach the further reaches where compressing structure decays, and where anomalies often lie.

Since the requirement was to force SUBDUE to analyze new regions of the graphs, the remedy was simply to delete all instances of the most-compressing subgraph from the traces. This encourages SUBDUE to discover dissimilar graphical features, compressing away graphical features of decreasing importance. As such, the method generates a natural, hierarchical derivation of process substructures in the form of a dendrogram. The dendrogram comprises the entire behavior of the log, with the ancestral components reflecting the most relevant and frequent graphical features of the log. This is amenable to anomaly detection since the less compressing a feature is, the more deviation it represents with respect to normative patterns and normal overall behavior, and hence will be placed lower in the dendrogram.

This gives the following process-oriented pattern-mining algorithm:

[algorithm text]

As shown, the Inductive Miner takes a workflow log and returns a process model, by which the traces can be re-generated as a collection of subgraphs. This is fed to SUBDUE to find the most compressing substructure, which is appended to the dendrogram before being deleted from all traces in which it occurs. This step repeats until no further progress can be made, when all traces have been compressed to their most elementary substructures. The dendrogram is returned, whose edges represent ancestry between compressing substructures and their constituent traces.

The success of this method lies in the dendrogram, as a descriptive model of the input log. The dendrogram can be analyzed in post-processing for common process features, redundant behavior, outliers, anomalies, and so on. Diamintini *et al* [13] have successfully detailed the uses for similar SUBDUE-based dendrograms, especially in the context of spaghetti processes cohering to no strict process definition.

This method of frequent-subgraph mining of workflow logs belongs to the family of dendrogram or tree-induction methods in process mining literature, and anomaly detection is one use among many for querying the dendrogram. For instance, while the low-frequency, outlier components of the dendrogram characterize anomalies, outliers, and noise, the ancestral components encode the most relevant substructures of the workflow log. Using this information, the process model returned by the Inductive Miner could be reduced to give a more concise process description. For instance, the underlying properties of an unstructured institution process could be discovered, and thereby the process could be identified, measured, and improved via business-process formalisms.

In this regard, coupling SUBDUE with the generalization feature of the Inductive Miner creates an extensible framework for more concise modelling of unstructured “spaghetti” process environments. Similarly, an analyst may examine highly similar components of the dendrogram, representing duplicate work or poor cohesion amongst business processes. Thus, the dendrogram extends to a range of pattern mining and other enterprise uses, beyond the scope of anomaly detection.

Anomaly detection lends an illustrative example in this context because of the structural characteristics of the dendrogram: given that anomalies are assumed to be infrequent events, subgraphs containing these will be among the last components compressed. The result is that for power-law distributed processes, the size of the dendrogram components decreases smoothly, then drops suddenly, such that the only remaining traces/subgraphs are those representing anomalies, outliers, or noise in the log.

Anomaly Detection Method [these are just notes for now; figure out where to insert]

Notably, the Bayesian interpretation was not fitted to the data/anomaly generation scheme, but rather is based on the target definition of anomalies as unusual activity occurring in the context of a normative pattern. Further, the Bayesian definition is more resilient to noise in this sense, which tends to occur in the vicinity of lower frequency structures…. etc

Algorithm Evaluation

Although real process-oriented datasets are available, they do not offer the controlled conditions sufficient to compare the characteristics of different algorithms. We instead opted to use a synthetic data generation algorithm similar to appendix A of [11], which was modified slightly to generate data directly from probability distributions embedded in the generated models. This approach generates random process models from which synthetic traces are generated, and thus the performance of an anomaly detection method can be assessed with respect to a known model. Likewise, to cohere to a stable performance baseline, we also used the same experimental parameters as described in Bezerra: 60 randomly-generated process models, 1000 traces [11].

Data Generation Algorithm

The data generation algorithm consisted of two steps: process model generation, and trace generation via these models. The first step was generating graphical process models, under a set of parameters describing the probability of generating various recursive structural features. These features include SEQ, OR-SPLIT, AND-SPLIT, and LOOP, defined as follows.

SEQ: the appending of a single activity.

OR-SPLIT: a single activity splits to one of two successors.

AND-SPLIT: a single activity splits to two parallel activities, both of which are traversed.

LOOP: An activity splits to a loop, then returns to that activity.

These recursive operators generate directed, potentially cyclic graphs of arbitrary complexity, with the constraint that the graph start at a single START node and all paths eventually end at a single END node. Additional complexity results from the fact that the set of “activities” includes the null transition, and as such the split constructs may divert to more than two activity paths, and so on. The parameters constrain model complexity to a probability distribution over these operators.

A second parameter set defines the trace-generation distribution constraining the graphical walks of traces, which is defined for OR-SPLIT and LOOP, the only operators with choice behavior. To ensure maximum partial-order entropy, activities lying within the same timestep are shuffled to achieve uniform random.

Lastly, some basic tests are applied to ensure sufficient complexity, probabilistic model-generation allows for the possibility of unusual models. These include verifying the model contain a minimum START-END path length, maximum number of anomalies, maximum number anomalous edges within an anomaly (to constrain anomaly size), minimum number of unique activities, and minimum number of paths from START to END.

In this manner, the models output by this method are thereby guaranteed to achieve sufficient complexity and to generate an exponential distribution of unique traces.

Noise Generation

The model and trace generation methods are sufficient to generate spaghetti-like process data in terms of model complexity via and activity complexity via , but remain constrained to the given model. For example, under this data generation method there will be no log confusion as to whether anactivity occurs before another if it always lies upstream in the paths contained in the model; hence, the underlying model strongly constrains the generated data. Although likely unusual, one might expect unstructured work environments to inevitable lead to occasional repetitions, whereby sub-components of some model are repeated or activities do not always follow one another. Such a scenario occurs, for instance, if prior to some software release a bug is detected after final testing, causing components of the testing activities to be repeated arbitrarily with respect to the software development process.

The simplest and perhaps worst-case way to simulate this is to randomly inject activities into the traces after trace generation. For every activity in each trace, with probability a random activity is inserted. The chosen activity is selected at uniform random from the set of all activities defined within the log. This method of noise addition is necessary to test our approach’s resilience to noise, and is justified since it creates many data outliers that would be too easily declared anomalous by an anomaly detection method that does not distinguish outliers from anomalies. Note that even small values of tend to create highly obfuscated partially-orderings for a process mining algorithm to mine: for a log of 1000 traces, for which the average trace length is 25, an value of 0.1 will have an expected number of “noise” activity occurrences of 0.1 \* 25 \* 1000 = 2500. This is an extreme number of random activities, since even a single misplaced activity with respect to the underlying model, breaks the logical rules and heuristics applied by most process mining algorithms to reproduce the underlying process model.

Experiment

To test our method, we desired datasets achieving “spaghetti” like behavior described previously. For each of 60 models, 1000 traces were generated for fixed and . As [BEZERRA], we evaluated the method in terms of accuracy, precision, recall, f-measure, and the f-measure, averaging these performance metrics over the 60 test logs.

The results were as follows…

Additionally, we wished to test our method by incorporating noise. This was done by varying the noise parameter , setting it to {0.0, 0.01, 0.05, 0.1, 0.2}.

Results were as follows…

Conclusions and Future Work

The method presented here demonstrates the desirable qualities shared by any anomaly detection approach: strong normative pattern definitions and statistical boundaries between anomalies, outliers, and normative patterns. This method has been shown to satisfy both requirements, strongly distinguishing anomalous traces, and generating normative patterns by which other process mining tasks can be performed. Notably, these properties satisfy a range of other process and monitoring tasks, as well as other process and graph mining tasks in fields such as biology, pharmacology, and chemical interaction.

The drawback to this method is its specialized noise-tolerance, a common problem faced by mining algorithms. While SUBDUE can find graphical patterns in an unsupervised manner, they become the only patterns by which the log is further compressed; such that even small deviations to the normative pattern are ignored, and may be flagged as anomalies later. From an anomaly-detection perspective, this strongly discriminatory behavior is desirable. On the other hand, from a process mining perspective, the goal is sometimes to mitigate such strong discrimination and to discover process models with a balanced, noise-tolerant tradeoff between specificity and generality. This yields the recurring discussion on specificity-generalization tradeoffs, to which all process mining approaches are subject. Future work lies in making the approach more noise tolerant, as the GBAD system determines acceptable deviations in the local context of a normative pattern discovered by SUBDUE using graph distance metrics.

References

[1] M. Dumas, W.M.P. van der Aalst, and A.H.M. ter Hofstede. Process-Aware Information Systems: Bridging People and Software through Process Technology. Wiley & Sons, 2005.

[2] Leemans, S., Fahland, D., van der Aalst, W.: Discovering block-structured process

models from event logs containing infrequent behaviour. In: Business Process Man-

agement Workshops. pp. 66–78 (2013).

[3] L. B. Holder. Empirical Substructure Discovery. In *Proceedings of the Sixth International Workshop on Machine Learning*, pages 133-136, 1989.

[4] Petri, Carl Adam; Reisig, Wolfgang (2008). "Petri net". Scholarpedia. **3** (4): 6477. doi:10.4249/scholarpedia.6477.

[5] Murata, Tadao. "Petri nets: Properties, analysis and applications." *Proceedings of the IEEE* 77.4 (1989): 541-580.

[6] Eberle, W. and Holder, L., Anomaly Detection in Data Represented as Graphs, *Intelligent Data Analysis, An International Journal*, Volume 11(6), 2007.

[7] Wil M.P. van der Aalst. *Process Mining Discovery, Conformance and Enhancement of Business Processes.* Heidelburg, Germany: Springer, 2011.

[8] Noble, Caleb C., and Diane J. Cook. "Graph-based anomaly detection." *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2003.

[9] Diamantini, Claudia, et al. "Pattern discovery from innovation processes." *Collaboration technologies and systems (CTS), 2013 international conference on*. IEEE, 2013.

[10] Genga, Laura, et al. "Subgraph mining for anomalous pattern discovery in event logs." (2016).

[11] Bezerra, Fábio, and Jacques Wainer. "Algorithms for anomaly detection of traces in logs of process aware information systems." *Information Systems* 38.1 (2013): 33-44.

[12] Bezerra, Fábio, Jacques Wainer, and Wil MP van der Aalst. "Anomaly detection using process mining." *Enterprise, business-process and information systems modeling* 29 (2009): 149-161.

[13] Diamantini, Claudia, Laura Genga, and Domenico Potena. "Esub: Exploration of subgraphs." *Proceedings of the BPM Demo Session* (2015): 70-74.

[14] N. Russell, A.H.M. ter Hofstede, W.M.P. van der Aalst, and N. Mulyar. Workflow Control-Flow Patterns: A Revised View. *BPM Center Report BPM-06-22* , BPMcenter.org, 2006.

[15] Control-Flow Patterns. Retrieved on September 13, 2017, from http://www.workflowpatterns.com/patterns/control/.

[16] Eberle, W., & Holder, L. (2009, April). Graph-Based Approaches to Insider Threat Detection. In *Proceedings of the 5th annual workshop on cyber security and information intelligence research: cyber security and information intelligence challenges and strategies* (p. 44). ACM.

[17] http://www.xes-standard.org/

[18] Herbst, Joachim. "A machine learning approach to workflow management." *ECML*. Vol. 1810. 2000.