TOAD: Trace Ordering for Anomaly Detection

Florian Richter, Yifeng Lu, Ludwig Zellner, Janina Sontheim, and Thomas Seidl
Ludwig-Maximilians-Universität München, Munich, Germany
Email: {richter, lu, zellner, sontheim, seidl}@dbs.ifi.lmu.de

***Abstract*—Outlier detection is one of the most important tasks to keep your processes in control. Unawareness of critical anomalies can lead to exhausting expenses, hence, it is highly beneficial to treat process failures as soon as possible. However, anomalies are difficult to detect due to their rarity though they occur too often to neglect the necessity of its detection. Even if the detection problem is solved, the treatment of singular anomalies and the adjustment of the process based on each abnormal trace is tedious and costly regarding time and money. To increase the efficiency of later anomaly treatment, we propose a novel strategy to detect collective anomalies. However, this is not equivalent to anomaly clustering as a post-processing step. TOAD orders process instances by similarity and detects abnormal accumulations of deviating cases. These collections are abnormal due to their aggregated behavior. Assuming that similar deviations are caused by the same reason, the treatment of such an anomaly is more cost-efficient than the handling of deviating singletons. Applying TOAD to an event log yields a ranking of significant, temporally abnormal trace collections, that provide a baseline for further analysis.**

1. Introduction

Deviations exist in probably every ever so well-defined process. On the one hand, they can bare risks of non-compliance and frauds, which cause expenses for the process owner. On the other hand, revealing beneficial deviations helps to improve the process with innovative ideas. Regardless of these types, a process can only be protected or improved if deviations are detected.

Process deviations are entities in a process, mostly traces or events, that behave abnormal with regards to a baseline model. Without such a model, the task is more complex. Anomaly detection is the task of defining normality in a domain in conclusion with the declaration of any observation in the data which does not behave according to this normality. Anomalies are partitioned into three categories, that are point anomalies, contextual anomalies and collective anomalies [5].

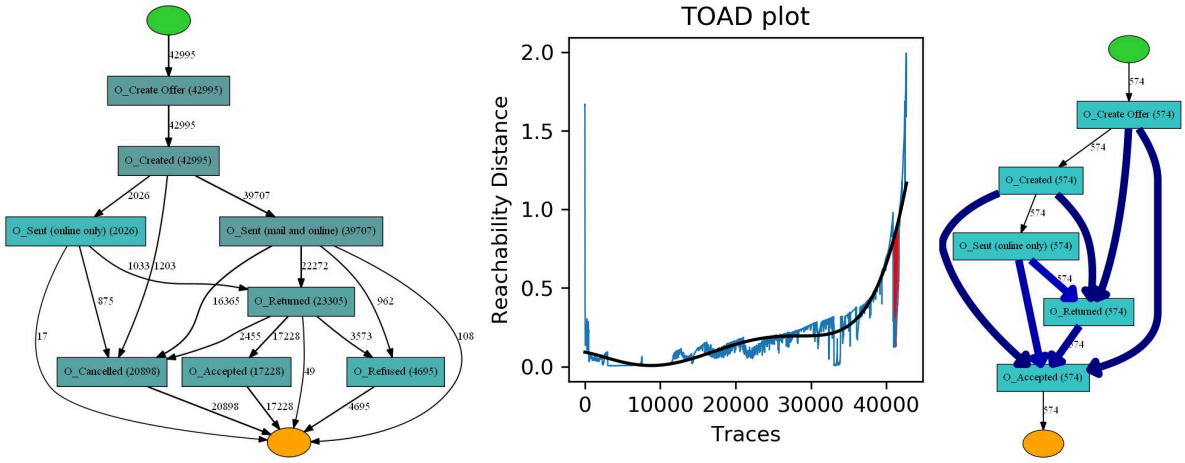
In traditional data mining, the search for point anomalies as outliers focuses on objects, that are prominently distinct in the sense of their behavior. The difference between an outlier and its most similar neighbor is typically very high [4]. This paradigm is useful in many scenarios, in which single objects have significant impact on the whole application. For example, a repair log contains reports from mechanics for a smart city bus fleet. If most buses succeed in a regular check-up, the ones that fail are point anomalies due to their individual failure. If such failures are depending on contextual factors, they call the anomaly a contextual anomaly. To continue with our example, some bus faults are season-sensitive. Air conditioning units, heaters or hydraulic systems are more likely to fail in summer and winter. Therefore, a faulty door in spring might have a different root cause than the same issue in winter.

However, in applications with extensive databases like process logs, where environments are continuously changing, it is not questionable if anomalies exists, but how to find them efficiently. Further, after the identification of anomalies, a solution for each issue is developed. This does not scale well for big data applications. In the end, it is a matter of efficiency and budget, which abnormal cases are treated and which have to be tolerated without any further actions.

In this work, we focus on collective anomalies. This paradigm ignores singular anomalies and yields clusters with abnormal behavior instead. The tagged abnormal traces might not be anomalies on their own, but their occurrence as a group is anomalous. Collective anomalies have been researched in the context of sequence data, spatial data and graph data. Especially for process data, this approach provides a strategy to efficiently handle anomalies from an economical perspective. Since it is very likely for an anomaly cluster that the root cause for the abnormality of all contained traces is the same, a common solution to the issue is applicable to the whole group. Returning to our example, a collective anomaly in the repair log reveals that some buses suffered from the same electrical short circuit, causing the hydraulic system to fail. Relying only on the previous paradigms, there would not have been an anomaly detected, since several buses had the same incident.

Our novel approach TOAD focuses on temporal deviations and utilizes temporal deviation profiles of cases, as presented in [16], to capture the temporal behavior of process instances. The concept of density is applied to define normality by the number of neighboring process instances. In [13], Tesseract showed that temporal outliers and drifts can occur even if the workflow of a process stays stable. However, Tesseract focuses only on drifts in singular activity relations. In this work, we extend the approach to the trace level and correlate activity relations of whole cases. Using the well-known cluster technique OPTICS on the temporal feature representation, we establish the name-giving Trace

978-1-7281-9832-3/20/$31.00 ©2020 IEEE
DOI 10.1109/ICPM49681.2020.00033



(a) Causal net for BPI Challenge 2017 Offer dataset. (b) TOAD plot with reference function. (c) Abnormal Submodel.

Figure 1. Process model of BPIC17 dataset, followed by reachability plot combined with a reference curve and the extracted anomaly process model.

Ordering for Anomaly Detection.

The main contributions of our work are (1) the adaptation of the temporal deviation signatures to generate a null hypothesis for an activity relation dependant density estimation, (2) transferring the concept of density to the process domain using OPTICS to identify micro-clusters of traces with higher relative density as anomalies. Further, we provide (3) a visual indication layer for process models to highlight and represent temporal deviation signatures.

2. Motivation

TOAD offers an explorative approach to identify collective temporal anomalies in process logs. Before we delve deeply into definitions and evaluations, we provide a real-world example to illustrate that collective temporal anomalies actually exist. The BPI Challenge 2017 dataset, which is described in detail in the evaluation, contains events of a load application process. In Fig. 1a we show the discovered causal net, yielded by the Heuristic Miner.

We apply TOAD, which identifies the most significant collective temporal anomaly. The yielded plot in Fig. 1b shows all traces on the x-axis, which are ordered according to their next neighbor, and the corresponding reachability distance, that is related to the local density of each trace. Therefore, a cluster of higher density is represented as a trough in this plot. The plot offers a simple and quick way to get an explorative view on the cluster structure in the dataset. We continue with automatically scanning for troughs with high relative troughs using a reference interpolation, plotted as a red line here. The troughs represent collections of traces, which are uncommonly more similar with respect to the expectation based on the remaining dataset. Depending on noise and the used parameters, many troughs are shown in the plot, however two very distinct ones are observed. We consider only the last one around $x = 40000$ due to its relative depth.

For its illustration, we mine a causal net only for the corresponding sublog in Fig. 1c. We add colored arrows as deviation indicators. The thick dark blue arrows show an accelerated execution of all events in this case in comparison to all other cases. These temporally abnormal cases contain only accepted loans. However, the delays indicate changed conditions in those cases, but we do not intend to theorize about the root cause here.

A deeper analysis is not possible without a domain expert and also not in the scope of this work. However, collective temporal anomalies exist in real-world process logs and due to their characteristics, traditional point anomaly detectors are not able to identify them. The model in Fig. 1c is a valid submodel of the original process model in Fig. 1b. Hence, anomaly detection based on the workflow would also miss this anomaly.

3. Related Work

In the area of clustering, especially trace clustering, some works have investigated different clustering techniques to aggregate process instances. The first established methods focus on vector space embeddings and map features of the sequences to points in a vector space. Basic approachs use bag-of-activities embeddings, which ignore the sequential information and represent traces as frequency vectors. Attached attributes like resources or lifecycle data can also be embedded into the vector space. This allows to use the full range of clusterings operating on vector spaces like k-means.

The vector space embedding of traces was first explored by Greco et al. [10]. Song et al. [15] used trace profiles which extend the previous idea and allow additional data to be captured. The number of attributes is not restricted such that complex behavior can be clustered using high-dimensional feature vectors. Ferreira et al. [9] clustered traces by building first-order Markov models for each cluster with the expectation-maximization technique. In [2] Bose

et al. introduced a specialized edit distance on traces and proposed an agglomerative hierarchical clustering approach. They also developed an approach [3] using subtraces of arbitrary length instead of n-grams to represent traces as feature vectors.

These techniques usually yield a partitioning of traces. Identified clusters provide a high intra-similarity and a low inter-similarity. These techniques can be summarized as passive trace clustering. Active trace clustering approaches, which were firstly introduced by De Weerdt [7] as ActiTraC, use an active learning process to develop trace clusters by taking the process discovery perspective into account. Traces are not mapped into a vector space but are greedily aggregated to clusters until the fitness of the model for this cluster drops. Sun et al. [17] also followed the hierarchical clustering method with compound trace clustering CTC. They developed a top-down trace clustering splitting logs in sublogs such that the complexity of each sublog matches the average complexity. The log with the highest model complexity is then split again. k-process [14] was developed by Richter et al. and uses process discovery and conformance checking in a k-means like procedure. Trace cluster centroids are built as process models and the assignment is performed using conformance checking as a distance to determine the most fitting cluster.

All these approaches work well to identify prominent process variants. However, we do not aim at finding trace clusters that exist on a common level. The deviations that our novel method reveals, are not mature clusters but anomalies. They can eventually grow into larger subprocesses and proper variants as well, but in the current state, the presented methods do not detect the minor deviations.

In contrast to clustering, traditional outlier detection methods focus on the identification of singular objects that deviate from the majority. The motivation for anomalous singleton traces is very intuitive: Revealing traces that do not behave as demanded, resulting from fraud or application failures. There is a manifold of different classification methods to solve this task in different applications. We can not cover the whole field, but few recent techniques shall be mentioned. Nolle et al. [12] utilized autoencoders to detect anomalies in business processes by training the process behavior on potentially noisy datasets and without prior knowledge. In [11], Myers et al. propose a method to identify anomalous behavior and cyber-attacks in industrial control system logs using conformance checking. Darmawan et al. [6] developed a method to filter a log, such that outliers are discarded and discovery can be applied on a clean anomaly-free database. All those methods amongst many others put emphasis on singular outliers. However, our novel approach aims at anomalous micro-clusters, so it is not a direct competitor for those methods. Our method is, as a result of its design, not able to detect those singularities and a direct comparison is not suitable. To the best of our knowledge, the perspective we take with our novel method, has not been investigated before in the field of process mining.

4. Preliminaries

Now, we will give the baseline definitions we are using regarding process mining and density-based clustering. Starting with the resources, the input data for TOAD are process event logs. An event is an observed witness of a process action. It contains the information about what activity happened at which timestamp in which case context at least, but it can provide additional data attributes. Here, we refer to the set of possible activities by A. An event $e = (c, a, t) \in (\mathbb{N} \times A \times \mathbb{N})$ is a tuple consisting of a case identifier c, an activity a and a timestamp t. Each event describes the execution of a certain activity corresponding to the case c at time t. We call a multiset, that contains multiple events over the same activity domain, a log L. Semantically, an event is a unique occurrence of an activity. Nevertheless, it can happen that the same case contains repetitions of a particular activity. If such events occur consecutively in a very small time frame, they might share the same event representation. Further, we call tuples of two activities $(a_1, a_2) \in A^2$ a relation.

A case c contains only the subset of a log, such that events in this subset share the same case identifier. As case identifier and cases themselves represent the same object, we will use it bilaterally. A trace is the projection of a case to the sequence of activity labels. However, in the literature, both terms are not used decisively and are often used in mixed contexts. Since our method puts a strong focus on the temporal view, it should be kept in mind, that we always consider the timestamps as a part of a trace in this work.

Continuing with density-based structuring, we require a measure of density. As a brief reminder, a distance is a positive-definite function, that is symmetrical and fulfills the triangle inequality. We do not go into detail here and for all mentions of a distance in the following, we are using the Euclidean distance, if not stated otherwise.

Density-based clustering is based on the concept of density, which is defined by two parameters: The radius ε to define the neighborhood of each object $N_\varepsilon(o)$ and the minimal number of points $MinPts$ required for a dense neighborhood.

One of the most popular density-based methods is DBSCAN [8]. It selects points and classifies them depending of their neighborhood as core, border or noise points. For a more in-depth description, we point to the corresponding work of Ester et al. A major drawback of DBSCAN is the difficulty to choose an appropriate value for the neighborhood distance ε. To overcome this issue, Ankerst et al. developed OPTICS [1]. Given $MinPts$, this method determines for each point its core distance, the minimal distance needed such that the ε-neighborhood contains $MinPts$ many points. For a point p let kNN be the k-th nearest neighbor and d a distance function. With $k = MinPts$, the core distance is defined by

$$\text{core}_{\varepsilon, MinPts}(p) = \begin{cases} d(p, \text{kNN}), & |N_\varepsilon(p)| \geq \text{MinPts} \\ \text{undefined}, & \text{otherwise} \end{cases}$$

The ε value is used as an upper bound for performance improvement. Using the core distance, the reachability distance for two objects o, p is defined as

$$\text{reach}_{\varepsilon,MinPts}(o,p) = \begin{cases} max(\text{core}_{\varepsilon,MinPts}(p), d(p,o)), & \text{if } |N_\varepsilon(p)| \geq \text{MinPts} \\ \text{undefined}, & \text{otherwise} \end{cases}$$

The set of data points gets ordered by its reachability distance. For each point, its successor is the point with the smallest reachability distance out of the unprocessed points. This ordering is not unique, due to start point ambiguity and potential choices between equidistant objects. Finally, a reachability plot is provided using the ordering on the x-axis and the reachability distance on the y-axis. Since dense object clusters in the data space have low pairwise reachability distances, they are accumulated in the plot and the cluster is identified as a trough in the reachability plot.

For a clustering analysis, the OPTICS plot is used to determine an ε-level, such that as many troughs as one likes are separated under a horizontal line. With this technique, clusters of equal densities can be identified in an exploratory task. However, our goal is different. The result of our novel method does not reveal distinct trace clusters of similar densities. We provide small clusters with anomalous densities relative to their neighborhoods as a result. In the following section, we go into detail to describe our changes to the adapted clustering technique, such that it becomes suitable to identify the desired results.

5. TOAD: Trace Ordering for Anomaly Detection

Our novel algorithm TOAD comprises of three major steps to provide anomalous outlier micro-clusters. First, we explain the mapping into a data space that keeps track of all temporal deviations. Second, we order the traces using OPTICS, based on nearest-neighbors. In the third part, we detect anomalies by analyzing the ordered plot and extract trace candidates with strong deviation characteristics.

5.1. Temporal Deviation Signatures

In [13], the authors focus on temporal concept drift detection regarding singular relations only. We adapt the idea of standardization via z-scoring, as it proposes a suitable representation of deviations. However, we extend the perspective from event level to traces.

Starting with a process log, we extract all event pairs sharing the same case identifier. Considering all relation candidates we determine the relation duration set RD for a given relation (a_1, a_2) as follows:

$$RD(a_1,a_2) = \{t_2 - t_1 \mid e_1, e_2 \in L \wedge e_1 = (c, a_1, t_1) \wedge e_2 = (c, a_2, t_2) \wedge t_1 < t_2\}$$

This set contains all durations between two particular activities, that are correlated in common process instances. We extract the arithmetic mean and standard deviation of all those relation duration sets, where $n_{(a_1,a_2)} = |RD(a_1, a_2)|$.

$$\mu_{(a_1,a_2)} = \frac{1}{n} \sum_{\delta_t \in RD(a_1,a_2)} \delta_t$$

$$\sigma_{(a_1,a_2)} = \sqrt{\frac{1}{n} \sum_{\delta_t \in RD(a_1,a_2)} (\delta_t - \mu_{(a_1,a_2)})^2}$$

In the next step we perform a z-scoring, which deals with unstable activities by down-weighting them in comparison to activities with small standard deviations. Z-scoring provides a standardization for the duration values, so we can align short-term and long-term activities in the same model. According to the work in [16], we derive temporal deviation signatures for each case. A temporal deviation signature is a matrix $TDS_{a_1,a_2} \in \mathbb{R}^{A \times A}$ that assigns the standardized duration to the activity relation, if the activity relation is present in the case:

$$(TDS_c)_{a_1,a_2} = \frac{|t_2 - t_1| - \mu_{(a_1,a_2)}}{\sigma_{(a_1,a_2)}}$$

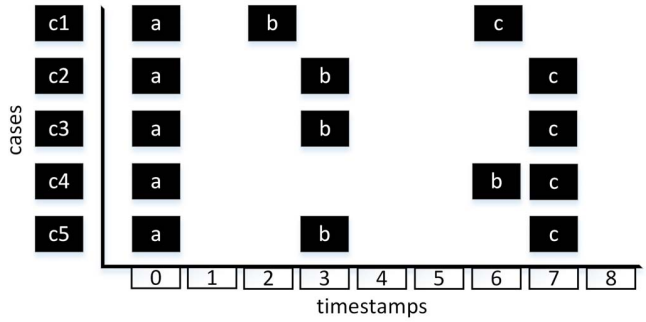
Otherwise, the value 0 is assigned.

For ordinary cases, which behave completely in-control, the deviation scores in the signature are next to zero. If we treat the signature as a vector in the linear space $\mathbb{R}^{|A|^2}$, those cases are represented as points around the origin, forming a sphere. Depending on the noise, the density of the sphere would follow the same distribution. For small datasets with few cases, Poisson-like characteristics towards the positive quadrants are likely, whereas larger datasets tend to form Gaussian-distributed density levels due to the central limit theorem.

We take a look at the example sketched in Fig. 2. All cases contain the three activities a, b and c and the majority require 3 time units between a and b and 4 time units between b and c. The first case $c1$ has a quicker execution and is therefore a deviation. The corresponding trace deviation signature is shown in Fig. 2b. The fourth case $c4$ shows also a deviation, as activity b is delayed, while the complete case execution stays close to the mean execution time.

5.2. Ordering Traces

In traditional data mining, a large manifold of clustering techniques have been proposed. We utilize density-based clustering as the next analysis step and adapt OPTICS [1] to perform the actual clustering. As we have explained before, ordinary cases will likely establish a sphere around the origin, if we treat the signatures as vectors. The maximum density is expected to be in the center. The more a singular case is deviating, the more it drifts towards sparser areas in the vector space representation. More interestingly, if a group of traces accumulates at a certain point, it establishes a denser area within sparsity. In the following we will show



(a) Traces of an example process, represented as Gantt chart.

	a	b	c
a	0.0	-1.0	-2.0
b	0.0	0.0	0.5
c	0.0	0.0	0.0

(b) Temporal deviation signature of case c1.

	a	b	c
a	0.0	1.9	0.5
b	0.0	0.0	-2.0
c	0.0	0.0	0.0

(c) Temporal deviation signature of case c4.

Figure 2. A tiny toy example demonstrating the derivation of trace deviation signatures.

how to automatically identify those collective anomalies or rare patterns.

DBSCAN [8] is a popular starting point for density-based clustering. It identifies clusters with similar densities and arbitrary shapes in the data space and can cope with noise. However, the density parameter is fixed so it can only identify clusters above a certain threshold level. Our data space varies in density and the more the temporal deviation signatures expand from the center, the sparser the anomalies get. The definition of anomalies does not rely on an absolute threshold, but on the surrounding space and its relative dependency gradient. As an extension to DBSCAN, OPTICS applies different levels of densities and creates a hierarchy of clusterings for a whole range of densities. The result is a 2-dimensional plot, that shows for each object, given on the x-axis, the density threshold necessary to be a cluster member. Since objects are ordered according to nearest neighbor distance, troughs in the plot depict dense aggregations of objects.

We establish a meaning of distance between two temporal deviation signatures S_{c_1} and S_{c_2}. Regarding distance measures, there is always the opportunity to dive deep into sophisticated measures. We will not focus on that here, but use the Euclidean distance. For two signatures S_{c_1} and S_{c_2}, the distance $\delta(S_{c_1}, S_{c_2})$ is defined as follows:

$$\delta(S_{c_1}, S_{c_2}) = \sqrt{\sum_{(a_1,a_2)\in A^2} (S_{c_1} - S_{c_2})^2}$$

Density is the central cluster criteria in this paradigm. It is basically the quotient of the number of traces in a certain area. For a signature S_c, we specify a distance threshold ε and count the number of other signatures $S_{c'}$ with a distance $\delta(S_c, S_{c'}) \leq \varepsilon$. The area within ε is called ε-neighborhood $N_\varepsilon(S_c)$. S_c is called dense or a core point, if its neighborhood contains more than $MinPts$ many other signatures for the predefined minimal point threshold $MinPts$. In other words, the distance for a neighborhood necessary to make a particular signature a core signature is given by

$$\text{core-dist}_{MinPts}(S_{c_0}) = \delta(S_{c_0}, S_{c_{MinPts}})$$

for the rising chain $\delta(S_{c_0}, S_{c_1}) < ... < \delta(S_{c_0}, S_{c_{|L|-1}})$. The definition is sound under the assumption that $MinPts \leq |L|$. Otherwise, the core-distance is undefined. Using the core-distance, we define the reachability distance between two signatures S_c and $S_{c'}$. This is either the core distance or the actual distance, whichever is greater:

$$\text{reach-dist}_{MinPts}(S_c, S_{c'}) = \max(\text{core-dist}_{MinPts}(S_c), \delta(S_c, S_{c'}))$$

Now, OPTICS orders all signatures. Starting with an arbitrary signature, all signatures are processed such that in each step, the signature with the smallest reachability distance to any previously processed signature is chosen as the next item. This way, closest points are processed with higher priority and compose a consecutive sequence in the process. The processing order defines the sequence of the final reachability distances to be selected, since only the distances between closest signatures are plotted. Hence, neighboring signatures are directly following in the reachability distance plot and the value of each signature defines the neighborhood size, such that both signatures belong to the same cluster. We will refer to this plot as the OPTICS plot.

5.3. Anomaly Detection

Traditionally, the OPTICS plot is used to visually identify clusters as valley-like structures. A density level is selected manually, such that the valleys under this level form valleys corresponding to the intuition. We propose an automated method, that does not reveal clusters with the same density, but anomalous aggregations of traces with deviating behavior instead. OPTICS applied to temporal deviation signatures usually yields a pattern, which assembles a large shallow valley for the majority of non-deviating traces, and which ascends into higher reachability distances while processing the deviating cases.

The OPTICS plot has indentations, that interrupt the otherwise smooth base-line of the analysis. The deeper such an indentation is, the denser is this small cluster of traces and the more similarity is shared between the affected traces. Hence, we compare the OPTICS plot with the smoothed base-line and focus on traces, that possess a large distance between smoothed and actual reachability distance to identify collective anomalies.

We generate the base-line by applying the Savitzky-Golay filter, a low-pass filter, to the reachability plot. It

can cope with the vast ascension in the outlier phase. Signal components of high frequencies are not discarded completely as in most other smoothing techniques. This advantage is achieved by using polynomial regression with low-degree polynomials over a small window around the particular point to be smoothed. The difference between original and smoothed curve is then used to identify deviations by applying peak-finding from the field of signal processing.

5.4. Limitations and Parameters

Now, we discuss some drawbacks of our method and their implications, as well as highlight the necessary parameters of TOAD. An important issue is the missing-of-values problem, which occurs by mapping traces to the temporal deviation profile space and apply a clustering method. Although all traces might be well-logged, traces usually contain only a subset of process-inherent activities. Hence, the temporal intervals are not defined for all relations in all traces. In our implementation, non-existing relations have a deviation score of zero. On the contrary, a zero is overloaded with the information, that the relation is present in the case, but does indeed show no deviation. Using the overloaded zero seems to be a naive but suitable solution. This causes a dense region at the zero, which is a false positive and can easily be discarded from the result set. Usually, anomalies that lack any abnormal behavior, are great to have but do not provide potential for improvement. However, we investigate other distance metrics in future works to find a better way for the comparison of 'incomparable' traces.

Regarding parameters, our method requires only few user-defined parameters. Since we adapt density-based clustering, the specification of density is a fundamental parameter. TOAD uses the number of minimum neighborhood points $MinPts$ to determine the trace ordering. For the resulting OPTICS plot, this parameter controls the granularity of the plot. A higher value increases the smoothness. We achieved good results with small values due to the following smoothing, e.g. $MinPts = 10$. After establishing the OPTICS plot, the Sawitzky-Golay filter needs the window size and the polynomial degree. As TOAD benefits from a very smooth baseline, we choose the window to be the maximum size possible. The polynomial degree should be an odd number. Usually the OPTICS plot starts low and raises when all denser points are processed and only singular outlier points are left. So for most distributions, the baseline is formed similar to a polynomial of degree 3 or 5. We suggest starting with those parameters. Later, the peak-finding requires to specify the width and prominence of a peak. Here, we suggest to use a top-k approach and start by analyzing high and large peaks first.

6. Evaluation

In the motivation section, we already show, that collective temporal anomalies actually exist. To the best of our knowledge, there are no other works that identify collective anomalies in the temporal perspective. So instead of a competitive analysis, we demonstrate the accuracy of TOAD by comparing results for different parameter settings and datasets. To conduct our experiments, we implemented TOAD as a stand-alone Python program that is available on Github[1].

6.1. Datasets

To evaluate the accuracy of an outlier detection method, problems for pure synthetic and real-world datasets arise. First, a synthetic dataset allows much freedom and it is very simple to design an artificial process that benefits the method to be evaluated. Anomaly detection has to be robust against noise. To show the robustness of TOAD, we need a dataset with an adequate level of realistic noise. It is difficult to claim that a synthetic dataset behaves realistically. Either the noise level is too low, which gives an advantage to the anomaly detection, or the noise is too strong and masks all anomalies. Therefore, a real-world dataset is a more suitable baseline, since the data obviously behaves realistically. However, we are mostly lacking a ground truth about which traces are anomalous. To overcome those issues, we mutate a real-world dataset by changing the duration of one activity. This simulates a group of traces, that might be more complicated and need additional time to process. We only mutate one activity at a time since more mutations make the abnormal cluster more distinct and simplify the detection.

We use the BPI challenge datasets from 2015[2] and 2017[3], in the following abbreviated as BPIC15 and BPIC17. BPIC15 contains data of building permit applications over four years in five Dutch municipalities. Five log partitions show the process of each municipality individually. Each sublog contains about thousand cases. The challenge of this dataset is not its number of cases. About 400 activities and therefore a large number of potential relations in all cases raise the complexity of the data.

In BPIC17, a loan application process of a Dutch financial institute over one year is logged. The offer log contains only a subset of 24 offer related activities. 128985 events are recorded in 42995 cases.

6.2. Experimental Setup

BPIC2015 contains a large activity domain. We identified the activity *phase decision sent* resp. *01_HOOFD_515* as a mandatory activity, that exists in every case. Although, this property is not a requirement for TOAD, it gives the opportunity to simulate a realistic and at the same time controllable deviation in all cases. We added a delay to this activity execution. The alteration of only one activity allows to filter infrequent activities and observe the impact of filtering while the detection challenge remains high.

1. https://github.com/Skarvir/TOAD
2. https://doi.org/10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1
3. https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b

Alterating multiple instances would simplify the detection as the distance to normal case profiles will rise. Since the remaining timestamps are unaltered, realistic noise still influences our results. In addition, other collective anomalies still exist in the data as well, so the result plot does contain further deviations.

For the BPIC2017 dataset, we manually identified a subset of cases which differ in their temporal execution. As already illustrated in Fig. 1c, this anomaly concerns accepted loan offers with an accelerated acceptance completion. The abnormal cases need on average 2.3 days to complete with a standard deviation of 1.9 days. In comparison, the remaining cases were completed in 13.5 days with 9.5 days as standard deviation. We applied TOAD on the first 5000 cases with different settings for MinPts. In addition, we used three different versions of the dataset. Version 1 contains all 5000 original cases. Version 2 contains only cases of length greater than 4 and Version 3 contains only accepted offers.

6.3. Results

Starting with the BPIC2017 dataset, we show in Fig. 3 the fitness and precision for the detection task to identify a collective anomaly as a classification problem. The measures should not be confused with process model quality, as it refers to classification accuracy. The anomaly consists of 135 cases, that represent accepted offers, but were more quickly accepted than on average. Roughly 1.5 standard deviations separates the abnormal cases from the remaining ones. In this setup, we compare the impact of the very important $MinPts$ parameter. For all three dataset variants, there is a clear drop in the fitness for $MinPts > 40$. Obviously, this value has to be smaller than the anomaly, that has to be detected. By choosing greater values, many smaller anomalies vanish in the result. However, the number of observed relations and therefore the number of dimensions increases distances between traces, so we do not detect the anomaly with a minimum of at least 40 traces within a cluster neighborhood.

The fitness drops slightly over the observed $MinPts$ values. This is explained by the fact that higher $MinPts$ values cause a smoother reachability curve and traces at the edges of a valley are not detected. The fitness for all three variants behaves similarly for increasing $MinPts$. For precision, matters change, as many cases contain only few activities and pairs of shorter cases tend to have smaller distances. Pruning of these short cases increases overall precision. As a rule of thumb, a narrow interval of case lengths is beneficial for the anomaly detection, as the dataset is more homogeneous. Otherwise, cases of short length are abnormal by size and are often detected first. The difference between all cases with length greater than 4 and only accepted cases is marginal, since the temporal representations differ in multiple dimensions. In the representation space, large traces exist in sparser areas and detection is made easier.

An important detail about the evaluation is the inequality of precision and fitness. Although the fitness drops significantly, the result is still very useful. We will illustrate this by applying TOAD on the modified BPIC2015 dataset. We inject abnormal traces by delaying the *phase decision sent* by a certain number of hours. On average, this activity follows its preceding event after about 6 days with a standard deviation of around 1 day. Due to the already existing temporal variance, the anomaly requires a delay of at least 8 hours to be detected. The precision is perfect for greater delays. We projected the log onto fewer activities, so the number of dimensions for the representation is reduced. Less dimensions increase the accuracy, which is not surprising. In the smallest case, only 6 activities are observed, leading to 30 relations, The third instance contains 11 activities, resulting in 107 relations. This dataset contains many unique activities or infrequent activities and a preprocessing for pruning is highly recommended.

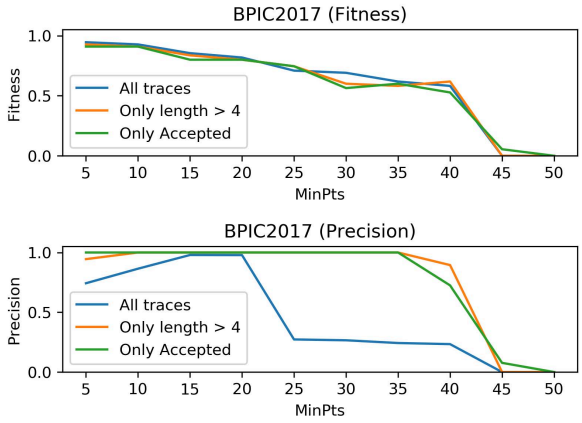


Figure 3. Accuracy scores of TOAD applied to the BPIC2017 dataset.

The fitness here is also not convincing by simply considering the numbers. We need to take a look on the actual result in Fig. 4. This result shows one of the more difficult evaluation instances. The reachability distance is plotted in blue, the reference curve is red. We highlighted the detected abnormal traces as a green area between the curves. All those detected traces belong to the ground truth, which leads to maximum precision. Taking a look onto the not detected abnormal traces, marked as yellow crosses, all but one trace are just besides the detected traces. TOAD is designed to identify a collective anomaly and the anomaly was correctly detected. Hence, the fitness value is not of the same importance as the precision value. Identifying and analyzing false anomalies costs much effort and time. Missing an abnormal trace is reasonable if the anomaly is detected. However, spending resources on chasing false anomalies should be avoided in many applications. If an actual detection algorithm for individual trace classification is needed, TOAD can still be used as a starting point and borders around the dense cores have to be investigated.

Summing up the results, the $MinPts$ parameter has to be chosen reasonable, since small values will lead to anomalies with few traces, while larger values cause most anomalies to disappear in the result. In the field of density-

based clustering, the research for good heuristics is still active. Also, pruning of short traces and narrowing the search space is very efficient.

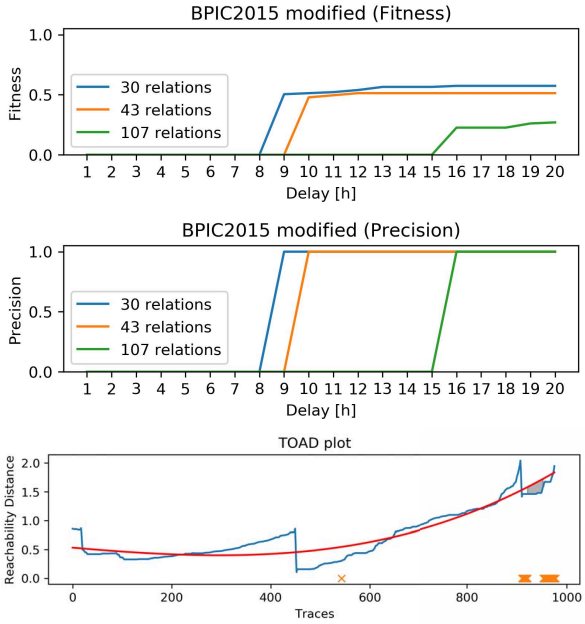


Figure 4. Accuracy scores of TOAD applied to the BPIC2015 dataset and TOAD result for the modified BPIC15, pruned to 107 relations with a 20h delay.

7. Conclusion

Identifying outliers is an important task for process improvement and failure prevention. In this work, we presented a method to detect micro-clusters with outlier characteristics. These process deviations as sublogs bear an economically efficient potential to search for process adaptations, either to include positive behavior or to avoid negative failures. Singular outliers are excluded in our method by design, as they require individual treatment and it is likely that their behavior does not occur a second time. Instead, our novel method TOAD puts emphasis on sets of traces, that possess a dense intra-density while being distant to other traces. Requiring only few parameters to specify and a low computational complexity, our method is well-suited for industrial applications.

In future works, we are planning to investigate the event stream capabilities of TOAD. A fast detection mechanism can help to detect outliers on-the-fly and call the attention of analysts to anomalies with a high impact due to the number of affected traces. In an operational support setting, the benefits of TOAD are even more beneficial, as analysis time becomes a scarce resource.

A further interesting direction adapts our approach and takes other attributes besides the temporal data into account. Weighting attributes of different specifications against each other like resources and time is a difficult task and more development is required to generalize TOAD to broader sets of data types. However, this branch is very yielding, because all sources of deviations are essential for a thorough root-cause analysis on micro-clusters.

References

[1] Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: Optics: ordering points to identify the clustering structure. ACM Sigmod record **28**(2), 49–60 (1999)

[2] Bose, R.J.C., Van der Aalst, W.M.: Context aware trace clustering: Towards improving process mining results. In: Proceedings of the 2009 SIAM International Conference on Data Mining. pp. 401–412. SIAM (2009)

[3] Bose, R.J.C., van der Aalst, W.M.: Trace clustering based on conserved patterns: Towards achieving better process models. In: International Conference on Business Process Management. pp. 170–181. Springer (2009)

[4] Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: Lof: identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD international conference on Management of data. pp. 93–104 (2000)

[5] Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM computing surveys (CSUR) **41**(3), 1–58 (2009)

[6] Darmawan, H., Sarno, R., Ahmadiyah, A.S., Sungkono, K.R., Wahyuni, C.S.: Anomaly detection based on control-flow pattern of parallel business processes. Telecommunication, Computing, Electronics and Control (TELKOMNIKA) **16**(6), 2808–2815 (2018)

[7] De Weerdt, J., vanden Broucke, S., Vanthienen, J., Baesens, B.: Active trace clustering for improved process discovery. IEEE Transactions on Knowledge and Data Engineering **25**(12), 2708–2720 (2013)

[8] Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Kdd. vol. 96-34, pp. 226–231 (1996)

[9] Ferreira, D., Zacarias, M., Malheiros, M., Ferreira, P.: Approaching process mining with sequence clustering: Experiments and findings. In: International Conference on Business Process Management. pp. 360–374. Springer (2007)

[10] Greco, G., Guzzo, A., Pontieri, L., Sacca, D.: Discovering expressive process models by clustering log traces. IEEE Transactions on Knowledge and Data Engineering **18**(8), 1010–1027 (2006)

[11] Myers, D., Suriadi, S., Radke, K., Foo, E.: Anomaly detection for industrial control systems using process mining. Computers & Security **78**, 103–125 (2018)

[12] Nolle, T., Luettgen, S., Seeliger, A., Mühlhäuser, M.: Analyzing business process anomalies using autoencoders. Machine Learning **107**(11), 1875–1893 (2018)

[13] Richter, F., Seidl, T.: Tesseract: Time-drifts in event streams using series of evolving rolling averages of completion times. In: International Conference on Business Process Management. pp. 289–305. Springer (2017)

[14] Richter, F., Wahl, F., Sydorova, A., Seidl, T.: k-process: Model-conformance-based clustering of process instances. In: Proceedings of the Conference on "Lernen, Wissen, Daten, Analysen", Berlin, Germany, 2019. vol. 2454, pp. 161–172 (2019)

[15] Song, M., Günther, C.W., Van der Aalst, W.M.: Trace clustering in process mining. In: International Conference on Business Process Management. pp. 109–120. Springer (2008)

[16] Sontheim, J., Richter, F., Seidl, T.: Temporal deviations on event sequences. In: LWDA (2019)

[17] Sun, Y., Bauer, B., Weidlich, M.: Compound trace clustering to generate accurate and simple sub-process models. In: International Conference on Service-Oriented Computing. pp. 175–190. Springer (2017)