OTOSO: Online Trace Ordering for Structural Overviews

Florian Richter(✉), Andrea Maldonado, Ludwig Zellner, and Thomas Seidl

Ludwig-Maximilians-Universität München, Munich, Germany
{richter,maldonado,zellner,seidl}@dbs.ifi.lmu.de

Abstract. Identifying structures in data is an essential step to enhance insights and understand applications. Clusters and anomalies are the basic building blocks for those structures and occur in various types. Clusters vary in shape and density, while anomalies occur as single-point outliers, contextual or collective anomalies. In online applications, clusters even have a higher complexity. Besides static clusters, which represent a persistent structure throughout the whole data stream, many clusters are dynamic, tend to drift and are only observable in certain time frames. Here, we propose OTOSO, a monitoring tool based on OPTICS. OTOSO is an anytime structure visualizer, that plots representations for density-based trace clusters in process event streams. It identifies temporal deviation clusters and visualizes them as a time-dependent graph. Each node represents a cluster of traces by size and density. Edges yield information about merging and splitting trace clusters. The aim is to provide an on-demand overview over the temporal deviation structure during the process execution. Not only for online applications, but also for static datasets, our approach yields insights about temporally limited occurrences of trace clusters, which are difficult to detect using a global clustering approach.

Keywords: Trace clustering · Visualization · Operational support · Anytime clustering

1 Introduction

The ongoing digitalization of industries and social systems creates a strong demand for analysis tools to transform data into useful insights. Especially early warning systems for already known issues or still uncovered problems are highly requested. However, without a thorough exploration of the data, those systems cannot be developed, since we need to know what we are looking for beforehand.

In online applications, the time for analysis is always very precious and never sufficient. Therefore, an in-depth analysis has to be postponed, as interesting and promising aspects have been identified. A more shallow high-level analysis is more suitable as a time-efficient first exploration.

In the field of clustering, DBSCAN [4] is a prominent technique for density-based clustering. However, finding good parameters to generate results that

S. Leemans and H. Leopold (Eds.): ICPM 2020 Workshops, LNBIP 406, pp. 218–229, 2021.
https://doi.org/10.1007/978-3-030-72693-5_17

leverage the data into a given story is very tedious. Restarting clustering algorithms with arbitrary parameters is very different from output-driven experimentation. Therefore, OPTICS [1] was proposed as an extension, that offers a two-dimensional visualization for any multidimensional dataset. In OPTICS plots, the structure of the data is abstracted and parameters for density-based clusterings are visually determined.

In an online process mining application, we need to increase the abstraction level even further. Anytime variants for DBSCAN and OPTICS have been proposed in literature already. However, the structure of an online process is not covered by observing an event stream and building an up-to-date process model. The time perspective provides clusters with a further dimension of volatility.

In the context of processes, we differentiate between the major behavior, the baseline process, and process variants with deviation behavior. During the process execution, the baseline stays mostly static and rarely tends to shift its behavior. In contrast, variants often traverse different lifecycles dynamically. They emerge at certain points in time, merge with other variants, separate again and disappear eventually. In some time intervals, variants can remain inactive and reappear seasonally or randomly later.

In this work, we propose OTOSO, an on-demand temporal structure visualization of event streams. It is based on OPTICS and developed to cope with dynamic structure transformations. OTOSO collects trace data from an event stream as temporal deviation signatures, generates temporary OPTICS plots and aggregates their information into a graph plot. This plot shows relations between baseline and variant clusters. In a quick analysis, structure changes are identified visually. Each cluster is represented as a node of a specific size at a point in time. Relations between clusters are indicated by edges between nodes. The whole plot can then be interpreted as a map, that show the dynamic changes of the process during the event stream.

2 Related Work

To the best of our knowledge, there is no direct competitor that proposes an anytime structure overview for event streams. However, there are related methods that have to be mentioned here. There is a plethora of published techniques regarding process discovery, conformance checking and clustering. Due to space constraints, we only mention works that have a focus on temporal perspectives or which work on event streams.

Event stream monitoring emerges as a required preprocessing step for anytime analyses. Works in this field mainly prepare intermediate data for process discovery [3,7,9] and conformance checking [2,13]. These works propose methods to analyze event streams, which is the more complex task in comparison to trace stream analysis. The latter paradigm assumes that events are already grouped into traces, which is mostly a difficult requirement. In many practical scenarios, there is also a strong concurrency between cases. Cases can become inactive or are stopped without any further information. An approach based on event streams has to come up with a heuristic to deal with the lack of information.

In the area of temporal anomaly detection, Rogge et al. [12] analyzed interim times between events by applying kernel density estimation to identify outliers in the temporal perspective. In [11], the authors identify such outliers of event pairs online by using hashing for event collecting and applying z-scoring to define an in-control area for unsuspicious event relations. In [10], this idea is leveraged on the trace level to detect collective trace anomalies using density-based clustering on temporal deviation signatures. We adapt the presented clustering technique for OTOSO.

The area of event stream concept drift detection contains more established works. In [6], Hassani elaborated the idea of [7] to detect work-flow-based concept drifts using different structural metrics on process models. In [8], the authors present a technique to change forecasting models according to changed environments due to concept drifts. However, we are not aware of any concept drift detection approaches taking the temporal perspective into account.

3 Preliminaries

An event stream $S : \mathbb{N} \to \mathbb{N} \times A \times \mathbb{N}$ is a mapping from natural numbers to the event domain. Each event $e = (c, a, t)$ consists of an case identifier $c \in \mathbb{N}$, an activity label $a \in A$ and a timestamp $t \in \mathbb{N}$. For case identifiers from another domain, there is typically a canonical translation into the natural numbers. The same holds for the timestamps. In the following, we will not distinguish between cases and case identifiers, as the context provides enough clarification.

Since OTOSO can also be applied to event logs, we define an event log as a finite multiset of events. Although an event log is mostly grouped by case identifiers, for OTOSO the log should be sorted by timestamp. Additional event attributes like resources are ignored in this work, although they might enhance the results in future works.

Next, we call tuples of two activities $(a_1, a_2) \in A^2$ relations. A relation (a_1, a_2) exists in a case c, if there are two events $e_1 = (c, a_1, t_1)$ and $e_2 = (c, a_2, t_2)$ with $t_1 < t_2$. We canonically define the mean μ and variance σ of all time intervals in a finite set of cases for a certain relation. Using z-scoring as follows we account for the imbalance between all different relations and define the temporal deviation signature as:

$$TDS^c(a_1, a_2) = \begin{cases} \frac{|t_2 - t_1| - \mu_{(a_1,a_2)}}{\sigma_{(a_1,a_2)}} & , e_1 = (c, a_1, t_1), e_2 = (c, a_2, t_2) \in c \\ 0 & , \text{otherwise} \end{cases}$$

In case of multiple occurrences of a relation, the average z-score is used. A distance is a positive-definite function, that is symmetrical and fulfills the triangle inequality. In the following, we use the Euclidian distance due to its popularity and will not go into detail about other functions in this work. For the clustering step, we require a measure of density. Density is defined by a number

of objects n in a certain area of radius ε. If an object, here a case represented by its temporal deviation signature, contains at least $MinPts$ many objects within a neighborhood $N_\varepsilon(c)$ of radius ε, this case is a core object. All cases within the neighborhood are at least border objects, if their neighborhood is not dense enough to be core objects themselves. All remaining cases are noise.

One of the most popular density-based methods is DBSCAN [4]. It selects objects 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 object its core distance, the minimal distance needed such that the ε-neighborhood contains $MinPts$ many objects. Derived from the core distance, the reachability distance between two objects is computed then. According to this distance, the processing order is depending on the nearest neighbor that has not been processed yet. This 2D reachability plot uses 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 clusters are identified as troughs in the reachability plot. Using a horizontal line as a density threshold, all troughs below this level represent clusters using the height as the according ε-value.

4 OTOSO

OPTICS visualizes the cluster structure of a static dataset. However, especially in process mining, process behaviors are dynamic and cluster structures are likely to change. To visualize not only a snapshot in a particular time frame, but the evolution of process variants and trace anomalies, we propose OTOSO, which is briefly summarized a visual time series of trace cluster structures. OTOSO consists of two phases. First, the event stream is observed and the necessary statistics are collected. By using a hashing data structure, the data is provided for the second module on-demand. At any point in time, the stored data can be queried as input for OPTICS to produce the current temporal cluster structure in the recent event stream. All those individual clustering snapshots are used to iteratively plot the clustering overview for the whole event stream.

4.1 Monitoring Temporal Deviations

OTOSO uses an event stream as input. In contrast to trace streams, the events have to be collected individually before case statistics can be extracted. A major problem of stream input is that we can never be sure that a case is still active. Therefore, we need an aging mechanism to discard old cases without the certainty that they are canceled or just paused and will be continued later.

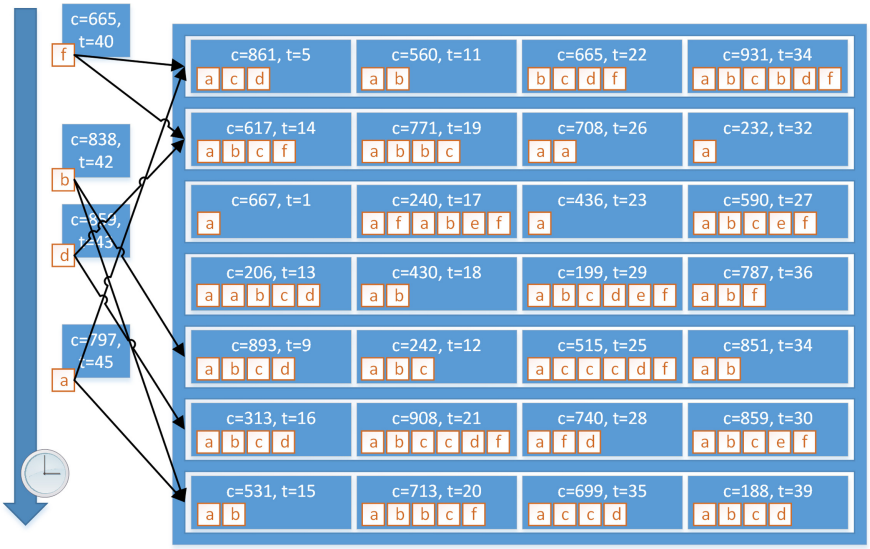


Fig. 1. Example hash table with $h = 7$ and $w = 4$. Each observed stream event has two potential rows to store it. Since the table is already full, either an event can be appended to its corresponding trace or an old trace has to be discarded. Do not be confused with the activity labels, since complete event information is stored.

We utilize Cuckoo-Hashing as it already provided a useful discarding technique for StrProM [7]. A hash table of height h is filled with case data, that is the last timestamp, the case identifier and all observed events. Two hash functions are applied on the case identifier to determine two potential hash table cells for each case. Instead of storing the case data directly in the hash table, we store a small and finite collection of cases in a cell. Technically, this width w of the table is implemented using arrays. Thus, the decaying factor can be adjusted without corrupting the operation complexity.

For each observed event, both hash functions are applied to identify all potential storage cells. If the case is already stored, it is updated by adding the event and setting the last-modified timestamp. In Fig. 1, the stream event in the top left corner belongs to case $c = 665$. A potential storage option is in the first hash table row. The case is already present in this row at the third position. We can update this cell by appending the event and updating the timestamp to $t = 40$. If the case has not been stored yet, we replace the case with the stored case, that has the oldest last-modified timestamp. The replaced case is the least recent one in this hash table cell. We try to insert it in the secondary position. Either, the secondary position has empty space, or we replace it again with the oldest case in this position. The procedure is recursively repeated until the secondary position has only more current entries and we discard the current item. Considering Fig. 1 again, the second stream event with $c = 838$ has storage options for row 5 and 7. Neither holds data for this case already. Using the first option, we

attempt to store this new case in the fifth row, depending on the first of both hash functions. The oldest case here is case $c = 893$. We replace it with the new case and try to re-insert $c = 893$. The timestamp $t = 9$ tends to be already deprecated, however, since there are older entries in the table, there might be a chance to discard it after a series of replacements. This would be the case, if the alternative storage position is in row 1 or 3. Otherwise, the already existing timestamps in the remaining rows are newer and case $c = 893$ is discarded.

With this strategy, the hash table is always a finite representation of the recent cases, however some older behaviors potentially survive in the data structure since the swap operations regard the table only partially. Another drawback is that events in the beginning of cases are represented excessively, as the chance to be discarded is increased for longer cases. Alternatively, the length of the case can be included in the discarding mechanism. Nevertheless, this gives older cases an advantage to be kept stored, since smaller and recent cases are discarded. To the best of our knowledge, a perfectly fair sampling for event streams is still an open research topic, so we accept the drawbacks and discard by recency only.

Regarding hash functions, there are various ways to implement a set of two functions. Most programming languages provide at least one built-in hash function. To derive a second one, it is mostly sufficient to reverse the case identifier and use the same function again. Another strategy splits the identifier in two chunks and uses the hash value for the first and for the second chunk to determine both positions. We did not perform an in-depth evaluation on this topic here.

4.2 Structure Analysis

The hash table provides at most $h \cdot w$ many cases at any point in time t. The cases do not have to be completed already. The complete hash table is processed to extract the case data and to generate the z-scored temporal deviation signatures for all cases, which is used as input for OPTICS to cluster the traces. The output gives an impression on the recent temporal trace clustering structure. For the stream structure overview, we extract all clusters depending on the chosen density parameters $(\varepsilon, MinPts)$. For each cluster C, we create a node at position $x = t$ and $y = \sum_{c \in C} \text{core}_{\varepsilon, MinPts}(c)$ which is the occurrence time and aggregated cluster density. The size of each node is depending on the number of contained cases in the cluster respectively the number of cluster elements that are currently stored in the hash table.

In the basic variant, OTOSO connects cluster nodes if the distance between cluster centers is below the distance threshold Δ_{TDS} and the nodes occur in consecutive time slots. The extension connects cluster nodes of distant time slots. This allows to identify temporally limited clusters that reoccur after a period of inactivity. In Fig. 2, OTOSO is applied to an event stream producing OPTICS plots for various timestamps. At a tickrate of $10k$ events, further intermediate results are requested. For four of these intermediate queries, we show the OPTICS plots in the top row of the figure. For each OPTICS plot, a vertical slice in the OTOSO plot below is generated. Typically, a process produces one

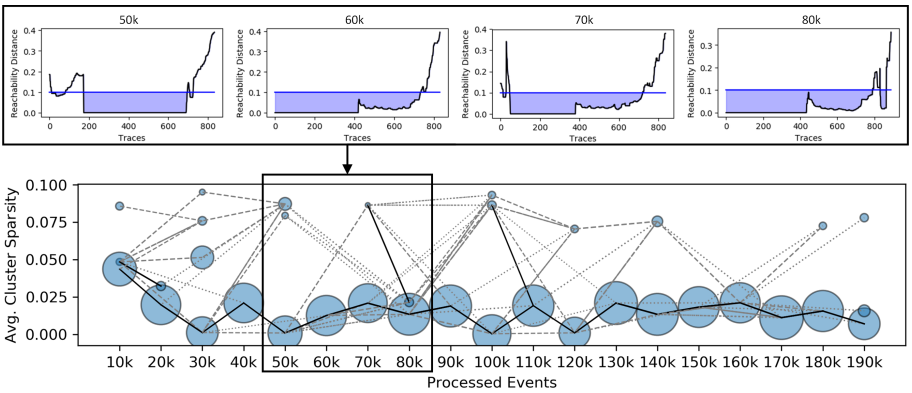


Fig. 2. OTOSO applied to an event stream. Each slice corresponds to a point in time and a hierarchy of clusterings at this timestamp.

major cluster containing cases that behave ordinary. These are the large spheres in each slice. For the $50k$ mark, besides the major cluster, two variants of low density are active. Both are related to previous queries, but disappear for the next two queries. Solid lines indicate a strong similarity between clusters of consecutive clusters. Dashed lines indicate similarity between slices over a larger timeframe. Here, we only include lines connecting slices within a timeframe of $30k$ events. In slice $60k$, all variants disappear. In $70k$, a small variant emerges. It has some similarity with the major cluster in $50k$, but no connection to the major cluster in $60k$. Hence, the temporal deviation profile first covered this deviation, but the variant did not occur in the succeeding process window. Interestingly, the small cluster in slice $80k$ grows slightly in size, but drastically in density. Regarding the solid line, we recognize a close similarity between both clusters, so their behavior represented by the temporal deviation signature is also similar.

This visualization allows to detect different structural changes in an event stream. Lifecycles of emerging and vanishing variants can be followed as illustrated before. The connections of a cluster node indicates, whether this variant has disappeared or has been inactive for some time. If a node emerges without initial connection, the corresponding variant starts suddenly. Otherwise, a connected new node hints towards a gradually emerging variant. These mechanisms are related to types of concept drift, however it is difficult to clearly label the effects according to sudden, gradual and incremental drifts due to the complexity of an event stream. Many activities and therefore activity relations are included in the temporal deviation profile. Nevertheless, the OTOSO plot gives an overview over the whole structure. A sudden drift, for instance, will likely affect a small number of traces and will maybe only affect some activities. The abstraction level of the visualization is to high to register concept drifts with a high confidence, except they appear as large-scale effects.

5 Evaluation

In the following, we evaluate the correlation between the size of the hash table and the currency of the collected event data. Afterwards, we show the benefits of applying OTOSO in comparison to using density-based clustering on the data as a static data chunk. Finally, we build a stream of a sequence of event logs to show the capability to detect the transitions between dissimilar event stream sections. We uploaded OTOSO into a GitHub project[1], thereby the experiments can be reproduced.

5.1 Datasets

Working with pure synthetic datasets causes some issues concerning the detection simplicity of anomalies or clusters in the data. We need datasets that are realistic, because synthetic datasets allow too much freedom and often are unfairly beneficial to the method's evaluation. Therefore, we utilize 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 partitions show the process of each municipality individually. Each sublog contains about a thousand cases. The challenge of this dataset lays in its about 400 activities and its resulting complexity from the large number of potential relations. The publications regarding this challenge show, that there is a high similarity between sublog 1, 2 and 5 while sublog 3 and 4 represent a slightly different behavior. 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. Due to its larger size, we are able to simulate an online observation of the whole fiscal year.

5.2 Hash Table Size

We use BPIC17 to investigate the influence of the hash table size on the currency of the data. Each event log is transformed into an event stream. Observing the stream event by event, each recent event is inserted into the hash table. Every 1000 events, we determine the average time difference to the current event timestamp. In Fig. 3, we show the results. Starting with a small hash table, which only contains 1000 cases, we compare three different dimensions for the table. In the first case, a table of height 10 with 100 buckets in each position is used. The second hash table has height 100 and width 10, while the third is a one-dimensional table of height 1000. The average recency is below 10 days. Towards the end, no new cases are starting, so no old cases are discarded and the table gets slightly outdated.

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

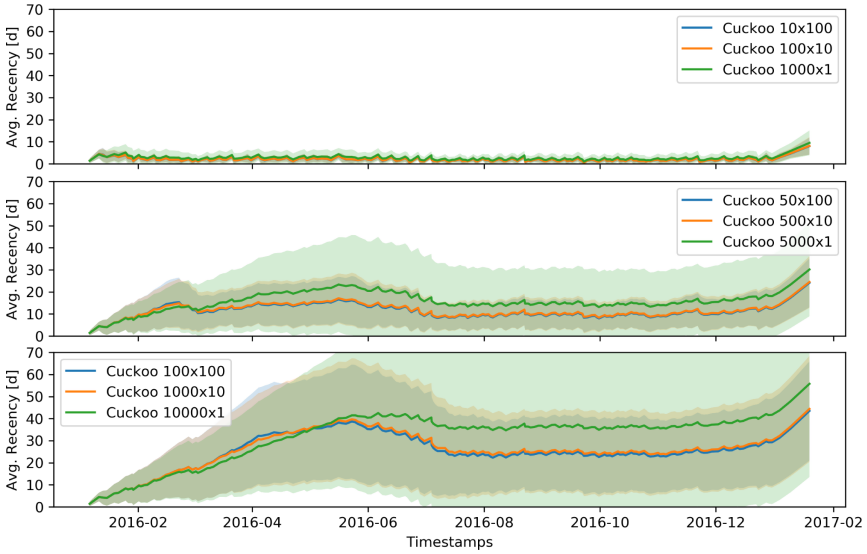


Fig. 3. Avg. recency and standard deviation is given for nine Cuckoo hash tables with different dimensions as $height \times width$.

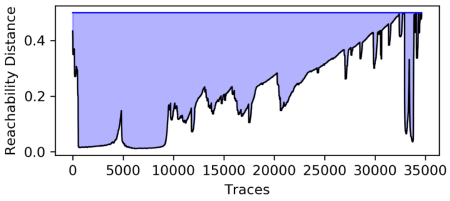
The second plot shows three hash tables of size 5000 having analogous changes regarding their dimensions. Due to the higher capacity, more cases can be stored and the table contains more obsolete items. Storing more items leads to a more stable clustering and following techniques are affected by noise or short-term outliers. There is no clear method to determine the best recency and the corresponding table size, since this is completely depending on the user-defined time window and the arrival frequency of events and cases. Finally, the application is also an important factor, since the detection of point-wise anomalies benefits from higher currency while the detection of long-term structures requires data with high stability. However, the important point we want to highlight is the advantage of using a two-dimensional hash table. The width allows shorter rehash cycles, which is already shown in [5,7]. The new insight here is the greater recency for small numbers of buckets in each position. Already in the second plot, but much clearer in the third one with a hash table of size 10000, the one-dimensional hash table has a delay of about 40 days, while both variants with few buckets have smaller temporal shifts. The difference between using 10 or 100 buckets is rather marginal. Therefore, we recommend using small numbers of buckets, since the iteration over a large list of buckets is more time-consuming than rehashing at another position.

5.3 Static Clustering vs. Dynamic Clustering

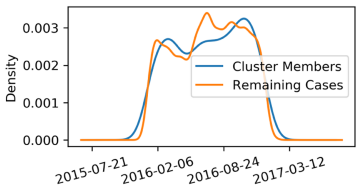
The BPIC17 dataset contains a significant cluster with deviating temporal behavior, that contains accepted offers with a delay in its execution. In Fig. 4a,

we show the result of OPTICS applied to the whole event log using the temporal deviation signatures as a representation. Using a neighborhood size of 0.5, two major clusters are yielded. The largest one contains the majority of cases and represents the baseline of this process. The second largest one is shown in OPTICS as a thinner and deep trough on the right side. Since this method yields a static overview over the temporal clustering structure, we would assume that the cluster is omnipresent during the complete event stream.

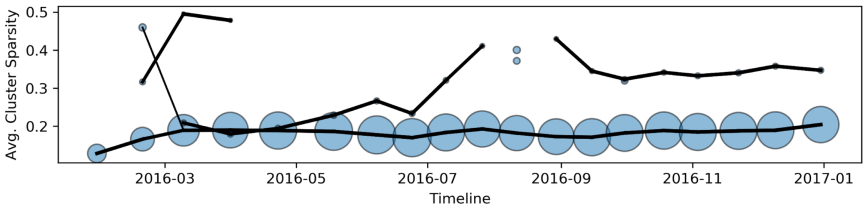
In Fig. 4c, the final OTOSO plot is given. After all events in the stream have been processed, the clusters are nodes with radii according to the number of contained cases. The height is determined by their density. Lines indicate a strong similarity between consecutive clusters. Thus, by following a line we observe the lifecycle of a specific cluster.



(a) OPTICS plot for BPIC17.



(b) KDE for BPIC17.



(c) OTOSO plot for the event stream of BPIC17.

Fig. 4. OPTICS and OTOSO applied on the BPIC17 datalog. $MinPts = 100$ and results are yielded each $10k$ events.

In the beginning, the results are not reliable. Many cases have been collected only partially yet. As a rule of thumb, we recommend to neglect insights from the first k cases if the hash table has size $k = h \cdot w$. Hence, starting with April, a baseline of large clusters has been emerged and retains an almost constant size for the remaining stream. More interesting is the other line above. It indicates a much smaller cluster, that still has a high density. During August the cluster vanishes but returns again in September. Instead, two new and dissimilar clusters emerge for this short period and vanish afterwards again. To show what OTOSO has highlighted there, we extract all cases contained in the previously mentioned deviating cluster. This set of cases corresponds to the thin and deep trough in Fig. 4a. For this cluster and also for the remaining cases, we plotted the starting

times as a kernel density estimation in Fig. 4b. Here, we observe a peak in starting cases in August. The rising number of arriving cases, which do not belong to the variant cluster, shifts more weight towards the baseline cluster and the two new variants. The resulting loss in density for our previous variant cluster leads to its disappearance for one observation tick. While it is possible to detect such effects with static methods, this analysis is quite tedious. Besides, we already knew what we were looking for. OTOSO highlights this anomaly during the online observation of the event stream. In applications, that require short reaction times, observing the OTOSO visualization provides a very quick indication for an abnormal behavior.

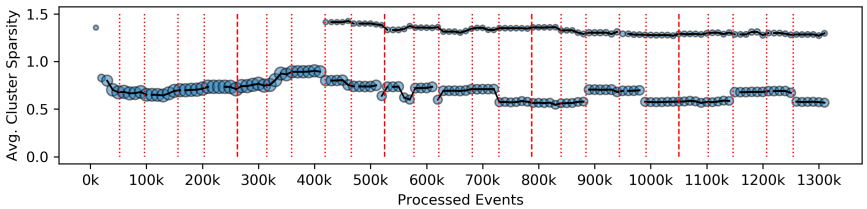


Fig. 5. OTOSO applied to a five-fold concatenation of all five BPIC15 sublogs. $MinPts = 100$ and an intermediate result is demanded every $10k$ events.

5.4 OTOSO on Event Stream with Concept Drifts

Finally, we use the BPIC15 dataset to how concept drifts affect the structural overview. The dataset is quite small, so we concatenate all five sublogs into one larger event log. Further, we concatenated this event log 5 times with itself to create an even larger log with five segments or 25 sublogs. This event log is then transformed into an event stream.

In Fig. 5, the OTOSO plot is given after processing the event stream. As discussed before, we neglect the results from the first two segments of the stream. After $500k$ events have been processed, the hash table is filled sufficiently and the structure of the data starts to appear. The red lines indicate the border points when a sublog ends and a new one starts. Especially in the last two segments, there is a significant similarity in BPIC15 between sublog 1, 2 and 5 and also between 3 and 4. The black similarity line indicates this relation. There is a much sparser and small cluster above. We do not have expert knowledge to verify or explain its meaning. On the one hand, it is possible to neglect it due to its sparsity. On the other hand, this cluster exists in all sublogs and it shows a strong similarity. In reality, we would recommend a thorough examination, but due to the lack of expert knowledge, we have to dispense with further speculations.

6 Conclusion

In a world of continuously emerging digitalization, it is very important to get preliminary insights early and with a high level of abstraction. OTOSO provides

an online overview over structures in an event stream. Emerging or vanishing clusters are visually identified and lifecycles of those structures are tracked.

Although some structural dimensions are monitored like density, size and similarity of clusters, process data contains more information, which can be used to augment the structural overview plot. Also, the plot depends on suitable user-defined parameters. Estimating good parameters is a very difficult task. Thus, and because a data stream cannot be replayed, it is beneficial to enable on-demand parameter adaptations while results are visualized.

References

1. Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: Optics: ordering points to identify the clustering structure. ACM SIGMOD Rec. **28**(2), 49–60 (1999)
2. Burattin, A., Carmona, J.: A framework for online conformance checking. In: Teniente, E., Weidlich, M. (eds.) BPM 2017. LNBIP, vol. 308, pp. 165–177. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-74030-0_12
3. Burattin, A., Sperduti, A., van der Aalst, W.M.: Control-flow discovery from event streams. In: 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 2420–2427. IEEE (2014)
4. 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, pp. 226–231 (1996)
5. Fan, B., Andersen, D.G., Kaminsky, M., Mitzenmacher, M.D.: Cuckoo filter: practically better than bloom. In: Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies, pp. 75–88 (2014)
6. Hassani, M.: Concept drift detection of event streams using an adaptive window. In: ECMS, pp. 230–239 (2019)
7. Hassani, M., Siccha, S., Richter, F., Seidl, T.: Efficient process discovery from event streams using sequential pattern mining. In: 2015 IEEE Symposium Series on Computational Intelligence, pp. 1366–1373. IEEE (2015)
8. Maisenbacher, M., Weidlich, M.: Handling concept drift in predictive process monitoring. In: 2017 IEEE International Conference on Services Computing (SCC), pp. 1–8. IEEE (2017)
9. Navarin, N., Cambiaso, M., Burattin, A., Maggi, F.M., Oneto, L., Sperduti, A.: Towards online discovery of data-aware declarative process models from event streams. In: 2020 International Joint Conference on Neural Networks. IEEE (2020)
10. Richter, F., Lu, Y., Sontheim, J., Zellner, L., Seidl, T.: TOAD: trace ordering for anomaly detection. In: 2020 International Conference on Process Mining (ICPM), pp. 1–8. IEEE (2020)
11. Richter, F., Seidl, T.: Looking into the tesseract: time-drifts in event streams using series of evolving rolling averages of completion times. Inf. Syst. **84**, 265–282 (2019)
12. Rogge-Solti, A., Kasneci, G.: Temporal anomaly detection in business processes. In: Sadiq, S., Soffer, P., Völzer, H. (eds.) BPM 2014. LNCS, vol. 8659, pp. 234–249. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10172-9_15
13. van Zelst, S.J., Bolt, A., Hassani, M., van Dongen, B.F., van der Aalst, W.M.: Online conformance checking: relating event streams to process models using prefix-alignments. Int. J. Data Sci. Anal. **8**(3), 269–284 (2019)