International Journal of Mechanical, Industrial and Aerospace Sciences
ISSN: 2517-9950
Vol:14, No:11, 2020

Object-Centric Process Mining Using Process Cubes

Anahita Farhang Ghahfarokhi, Alessandro Berti, Wil M.P. van der Aalst

Abstract—Process mining provides ways to analyze business processes. Common process mining techniques consider the process as a whole. However, in real-life business processes, different behaviors exist that make the overall process too complex to interpret. Process comparison is a branch of process mining that isolates different behaviors of the process from each other by using process cubes. Process cubes organize event data using different dimensions. Each cell contains a set of events that can be used as an input to apply process mining techniques. Existing work on process cubes assume single case notions. However, in real processes, several case notions (e.g., order, item, package, etc.) are intertwined. Object-centric process mining is a new branch of process mining addressing multiple case notions in a process. To make a bridge between object-centric process mining and process comparison, we propose a process cube framework, which supports process cube operations such as slice and dice on object-centric event logs. To facilitate the comparison, the framework is integrated with several object-centric process discovery approaches.

Keywords—Process mining, multidimensional process mining, multi-perspective business processes, OLAP, process cubes, process discovery.

I. Introduction

EVERY organization has to manage business processes such as the Purchase-to-Pay (P2P) and Order-to-Cash (O2C) processes. To manage real-life processes we need to consider two challenges. First, the nature of the business processes is not static due to the environmental changes (e.g., seasonal demands, changing in customer preferences). Thus, different behaviors may exist in business processes [1]. We can compare different process behaviors using process cubes. Process cubes are inspired by the notion of OLAP [2]. Similar operations to OLAP are defined for process cubes, i.e., slice, dice, drill-down, and roll-up [3].

Second, multiple objects interact with each other in the business processes [4]. Considering simple P2P and O2C processes, multiple objects, e.g., order, item, and package are involved. Moreover, different relations exist between these objects. For example, an order contains multiple items and multiple items are packed in one package for the delivery. Each of these objects can be considered as a case notion (i.e., a process instance). Therefore, in real business processes, extracted from ERP systems, we have multiple case notions. However, like most existing process mining techniques, current process cube approaches can only handle one case notion at a time [1], [5], [6]. Thus, they cannot cover multiple case notions involved in business processes such as orders and items. The interaction between multiple case notions can be analyzed using object-centric process mining [4]. This emerging subfield of process mining provides a more general vision of business processes by considering multiple case notions in processes.

Anahita Farhang Ghahfarokhi, Alessandro Berti, Wil M.P. van der Aalst are with the Process and Data Science institute, RWTH Aachen University, Germany (e-mail: farhang@pads.rwth-aachen.de, a.berti@pads.rwth-aachen.de, wvdaalst@pads.rwth-aachen.de).

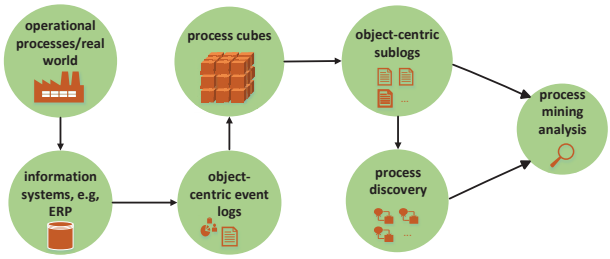


Fig. 1 Schematic overview of the proposed approach. In the proposed framework, we feed the process cube with object-centric event logs

In this paper, we address the problem of handling multiple interacting case notions in process cubes. In order to fit object-centric event logs into process cubes, some challenges arise due to the nature of object-centric event logs, where we need to consider multiple case notions. In traditional event logs, each event refers to a single case notion, an activity, a timestamp, and some possible additional attributes (e.g., location, cost, etc.). However, in object-centric event logs, each event may refer to multiple case notions. For example, suppose we have an event with a confirm order, related to one order but multiple items. In fact, there is a one-to-many relationship between order and item. Then, each event is related to one order and many items, i.e., convergence [4]. Considering slicing based on item dimension, the values of item dimension are not atomic. However, in the existing process cubes [1], [5], [7], [8], we need atomic values to apply each operation.

In this paper, we extend the notion of process cubes to be applicable to non-atomic values and provide an open-source implementation of our object-centric process cube approach. Moreover, to facilitate the comparison, we use object-centric process discovery approaches. We can extract process models of the cells of the process cubes and by comparing them against each other, we are able to do performance analysis from different angles.

A schema of the proposed approach is shown in Fig. 1. In the first step, we have operational and business processes running in the real world. The information of these business processes is recorded in information systems such as SAP or ERP systems. We can extract object-centric event logs, i.e., event logs with multiple case notions from data stored in information systems. Then, we use the object-centric event log as an input for the process cube. In the process cube, different operations such as slicing and dicing are applicable. Through the application of different operations, we obtain object-centric

sublogs. We can apply the object-centric process discovery methods on the object-centric sublogs to have the model of the process. Using such object-centric sublogs and the discovered models, we can analyze the process and find the pitfalls of the process.

The structure of the paper is as follows. In Section II, we discuss related work. In Section III, we describe the running example that is used throughout the paper. In Section IV, we extend the definitions of the process cube to support object-centric event logs. Section V provides our implementation of the proposed framework. In Section VI, we provide performance related results of our framework. Finally, Section VII concludes the paper and discusses the future work.

II. Related Work

In this section, first, we present the work related to object-centric process mining. Then, we discuss the developed approaches on process comparison.

One of the approaches developed to model the processes with multiple case notions is artifact-centric modeling. Artifacts combine process and data as the building blocks [9]. In [10], the authors formulate artifact-based business models and develop an initial framework that enables the automated construction of the processes. In [11], the authors introduce an artifact-centric process model, showing business objects and their life-cycles. The proposed techniques do not show the whole process in one diagram, which leads to losing a general vision over the process. Object-Centric Behavioral Constraint (OCBC) models show the whole process in one diagram and incorporate data perspective in the process model [12]. The main challenge for OCBC models is complexity, which leads to the development of MVP (Multiple Viewpoint) Models [13]. MVP models are graphs, annotated by frequency and performance information, where relationships between activities are shown by colored arcs. Object-centric Petri nets are another type of object-centric process models that can be extracted from object-centric event logs which provide the execution semantics [14]. In this paper, comparison between the MVP models and Object-centric Petri nets extracted from cube operations against the whole cube is possible, which is helpful in process analysis.

Process cubes are inspired by OLAP, where event data are organized using dimensions. Each cell contains events, which can be used as an input to apply process mining techniques such as process discovery. In [2], the event cube is introduced with the application of OLAP operations on event data. The first notion for process cubes was proposed in [6] and then enhanced in [1]. In [1] an approach for interactive analysis of the event data is also proposed. Process cubes were used for analysis in several case studies for different purposes [7], [5], [15]. Although, none of them addresses handling object-centric event logs.

III. Running Example

The example process is from the publicly available SAP IDES system which belongs to a real purchasing process (available in https://gitlab.com/Anahita-Farhang/process-cube). It contains 17500 events, 4 object types (i.e., case notions), 4 attributes, and 10 number of activities recorded from 2000 until 2020. A fragment of a the object-centric event log is shown in Table I. There is one-to-many relationship between order and item and one-to-one relationship between order and invoice. A part of the MVP model of the process is shown in Fig. 2. MVPs are graphs in which nodes represent activities and there is an edge between two nodes if there is at least one event in the event log where the source activity is followed by the target activity. In the MVP models, arcs with different colors represent different case notions [4]. Following the blue and purple arcs, the order and item go through create purchase order, and enter incoming invoice and a few of them leave the process at the *cancel invoice document*.

We also use Object-centric Petri nets in this paper. These discovery techniques focus on solitary processes. There may exist different behaviors in the processes. The variability in this purchasing process motivates us to extend the process cubes notion to support object-centric event logs to compare different processes.

IV. Object-Centric Process Cubes

In this section, we formalize the notion of object-centric process cubes. Using the running example, described in Section III, we provide examples for the object-centric process cube notion. A process cube is formed by its structure and an object-centric event log. The structure describes the distribution of the cells and the object-centric event log is used to materialize the cells of cube by events.

A. Object-Centric Event Log

The object-centric event log, shown in Table I, represents a collection of events and that is totally ordered based on the timestamp. Each event consists of an event identifier, attributes, and object types (i.e., case notions). To formalize the object-centric event log, we define the universes to be used throughout the paper.

Definition 1 (Universes)**.** *We define the following universes:*

- $\mathbb{U}_{ei}$ *is the universe of event identifiers,*
- $\mathbb{U}_{att}$ *is the universe of all possible attribute names,*
- $\mathbb{U}_{v}$ *is the universe of all possible attribute values,*
- $\mathbb{U}_{vmap} = \mathbb{U}_{att} \nrightarrow \mathbb{U}_{v}$ *is the universe of functions mapping attributes on attribute values,*

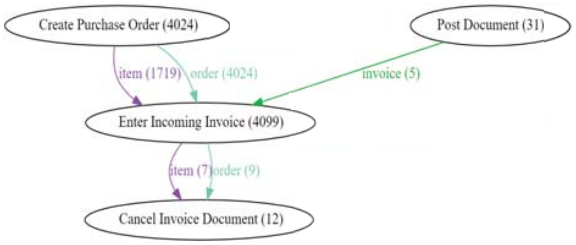


Fig. 2 MVP model for a fragment of the process

TABLE I
A FRAGMENT OF AN OBJECT-CENTRIC EVENT LOG

event identifier	attributes			object types		
	activity	timestamp	resource	order	item	invoice
0001	*create purchase order*	01-01-2000:00.00	USER01	$\{o_1\}$	$\{i_1, i_2, i_3\}$	{}
0002	*post document*	01-01-2000:08.05	USER01	{}	{}	$\{inv_1\}$
...	...	...	...	...	...	...
17499	*enter incoming invoice*	20-05-2020:15.11	USER50	$\{o_{12000}\}$	$\{i_{27005}\}$	$\{inv_{800}\}$
17500	*park invoice*	20-05-2020:15.14	USER42	$\{o_{12000}\}$	{}	{}

- $\mathbb{U}_{ot}$ *is the universe of all object types,*
- $\mathbb{U}_{oi}$ *is the universe of object identifiers,*
- $\mathbb{U}_s = (\mathcal{P}(\mathbb{U}_v) \cup \mathcal{P}(\mathbb{U}_{oi})) \setminus \{\emptyset\}$ *is the universe of value sets excluding the set containing the empty set,*
- $\mathbb{U}_h = (\mathcal{P}(\mathcal{P}(\mathbb{U}_v)) \cup \mathcal{P}(\mathcal{P}(\mathbb{U}_{oi}))) \setminus \{\emptyset, \{\emptyset\}\}$ *is the universe of value set collections excluding the set and the set collection containing the empty set,*
- $type \in \mathbb{U}_{oi} \rightarrow \mathbb{U}_{ot}$ *assigns precisely one type to each object identifier,*
- $\mathbb{U}_{omap}$ *is the universe of all the object mappings indicating which object identifiers are included per type.* $\mathbb{U}_{omap}$ *is defined like:*

$$\mathbb{U}_{omap} = \{omap \in \mathbb{U}_{ot} \rightarrow \mathcal{P}(\mathbb{U}_{oi}) \mid \forall_{ot \in \mathbb{U}_{ot}} \forall_{oi \in omap(ot)} type(oi) = ot\}$$

- $\mathbb{U}_{event} = \mathbb{U}_{ei} \times \mathbb{U}_{vmap} \times \mathbb{U}_{omap}$ *is the universe of events.*

$e = (ei, vmap, omap) \in \mathbb{U}_{event}$ is an event with event identifier ei, referring to the objects specified in $omap$, and having attribute values specified by $vmap$. Each row in the Table I refers to an event, which contains an event-identifier, attribute values, and object identifiers. Note that $\mathbb{U}_s$ does not include $\{\emptyset\}$ and $\mathbb{U}_h$ does not include $\{\emptyset, \{\emptyset\}\}$. These values are created after the power set generation. However, it is not meaningful for these sets to contain such values.

Definition 2 (Object-Centric Event Log)**.** *An object-centric event log is a tuple* $E = (EI, ATT, OT, \pi_{vmap}, \pi_{omap}) \in \mathcal{P}(\mathbb{U}_{ei}) \times \mathcal{P}(\mathbb{U}_{att}) \times \mathcal{P}(\mathbb{U}_{ot}) \times \mathcal{P}(\mathbb{U}_{vmap}) \times \mathcal{P}(\mathbb{U}_{omap})$ *where* $\pi_{vmap} \in EI \rightarrow U_{vmap}$ *is a function mapping each event to its attribute mapping, and* $\pi_{omap} \in EI \rightarrow U_{omap}$ *is a function mapping each event to its object mapping. Event identifiers are unique and time is non-descending.*

The first column in the object-centric event log of the running example shows unique event identifiers. Consider e_1, the first event in Table I. $\pi_{vmap}(e_1)(activity)$=*create purchase order*, $\pi_{vmap}(e_1)(resource)$=*USER01*, $\pi_{omap}(e_1)(order) = \{o_1\}$, and $\pi_{omap}(e_1)(item) = \{i_1, i_2, i_3\}$.

B. Process Cube Structure

We define the structure of the process cube independent from the object-centric event log. A process cube structure is fully specified by the set of dimensions.

Definition 3 (Process Cube Structure)**.** *A process cube structure is a triplet* $PCS = (D, val, gran)$ *where:*

- D *is a set of dimensions,*
- $val \in D \rightarrow \mathbb{U}_s$ *is a function associating a dimension to a set of values.*
- $gran \in D \rightarrow \mathbb{U}_h$ *defines a level for each dimension such that for any* $d \in D : val(d) = \cup gran(d)$.

A dimension d has a value $val(d)$ and a granularity $gran(d)$. The possible set of values for each dimension is specified by $val(d)$ and a subset of these values exist in a sample of the process cube. For example, $val(item) = \{i_1, i_2, i_3, ..., i_{27005}\}$ and $val(activity) = \{$*create purchase order*$, ...,$*park invoice*$\}$.
A dimension also has a granularity $gran(d)$ which is a set of sets. For example, $gran(timestamp)$ contains sets such as T_{2017} and T_{2018} each showing all the timestamps in a particular year. These sets form levels based on set inclusion (e.g., T_{2019} dominates $T_{Apr-2019}$).

The content of the cube is the object-centric event log. Therefore, we make the *process cube structure* and the *object-centric event log* compatible.

Definition 4 (Compatible)**.** *Let* $E = (E, ATT, OT, \pi_{vmap}, \pi_{omap})$ *be an object-centric event log and* $PCS = (D, val, gran)$ *be a process cube structure. They are compatible if:*

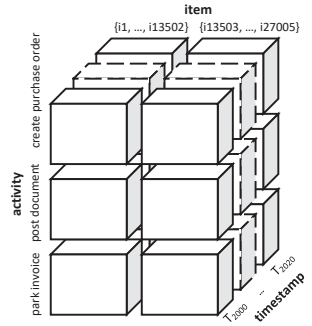
- $D \subseteq OT \cup ATT$*, dimensions should correspond to attributes or object types,*
- *for any* $d \in D \cap ATT$ *and* $e \in E : \pi_{vmap}(e)(d) \in val(d)$,
- *for any* $d \in D \cap OT$ *and* $e \in E : \pi_{omap}(e)(d) \subseteq val(d)$.

In making the process cube structure and the object-centric event log compatible, there is a difference between dimensions that correspond to object types and dimensions that correspond to attributes. This difference arises due to the non-atomic values for object types. Consider the activity as the dimension, which is an attribute, $activity \in D \cap ATT$, then $\pi_{vmap}(e_1)(activity) \in \{$*create purchase order*$, ...,$*park invoice*$\}$. However, if we consider item as the dimension, which is an object type, $item \in D \cap OT$, then $\pi_{omap}(e_1)(item) \subseteq \{i_1, ..., i_{27005}\}$.

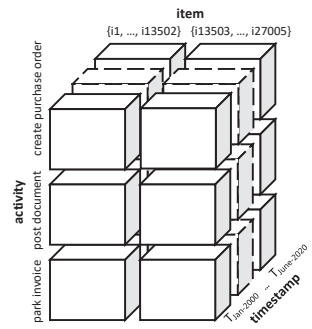
Different operations are possible in process cubes. By applying process cube operations such as slicing, the content of the process cube structure and object-centric event log do not change. We only change the fragment of the cube that we visualize. A process cube view defines the events that are visible for us.

Definition 5 (Process Cube View)**.** *Let* $PCS = (D, val, gran)$ *be a process cube structure. A process cube view is a pair* $PCV = (D_{sel}, sel)$ *such that:*

- $D_{sel} \subseteq D$ *are the selected dimensions,*



(a) A process cube view with the granularity of year for the time dimension



(b) A process cube view with the granularity of month for the time dimension

Fig. 3 Example of different process cube views of the same process cube structure

- *$sel \in D \rightarrow \mathbb{U}_h$ is a function selecting the part of the level considered per dimension. The function sel is such that for any $d \in D$:*
  - *$sel(d) \subseteq gran(d)$*
  - *for any $S_1, S_2 \in sel(d) : S_1 \subseteq S_2$ implies $S_1 = S_2$.*

A process cube view is a cube with $|D_{sel}| \leq |D|$ dimensions. Fig. 3a shows an example of a process cube view with three dimensions. The selected dimensions are activity, item and timestamp ($D_{sel} = \{item, timestamp, activity\}$). Function sel selects values for all dimensions regardless of whether they are in D_{sel} or not. For the $D \backslash D_{sel}$ we cannot see the values of sel in the *process cube view*. However, the values of sel exist for these dimensions and these dimensions may have been used in filtering. For example, in slicing the dimension is no longer visible but it is used in filtering. In the process cube view shown in Fig. 3a, $sel(timestamp) = \{T_{2000}, ..., T_{2020}\}$. Figure 3b shows another view where $sel(timestamp) = \{T_{Jan-2000}, ..., T_{June-2020}\}$. Through the requirement $S_1 \subseteq S_2$ implies $S_1 = S_2$, we ensure that elements of $sel(d)$ do not intersect, e.g., $sel(timestamp) = \{T_{Jan-2017}, T_{2017}, T_{2018}, T_{2019}\}$ is not possible. By having the process cube view and object-centric event log, we materialize the cells by events. We can extract event logs from cells of the process cube to apply process mining techniques such as process discovery.

Definition 6 (Materialized Process Cube View)**.** *Let process cube structure $PCS = (D, val, gran)$ and object-centric event log $E = (E, ATT, OT, \pi_{vmap}, \pi_{omap})$ be compatible. The materialized process cube for some view $PCV = (D_{sel}, sel)$ of PCS is $M_{E,PCV} = \{(c, events(c)) \mid c \in cells\}$ with $cells = \{c \in D_{sel} \rightarrow \mathbb{U}_s \mid \forall_{d \in D_{sel}} c(d) \in sel(d)\}$ being the cells of the cube and*

$$
\begin{aligned}
events(c) = \\
&\{e \in E \mid \forall_{d \in D_{sel} \cap ATT} \ \pi_{vmap}(e)(d) \in c(d) \\
&\wedge \forall_{d \in D \cap ATT} \ \pi_{vmap}(e)(d) \in \cup \, sel(d) \\
&\wedge \forall_{d \in D_{sel} \cap OT} \ c(d) \subseteq \pi_{omap}(e)(d) \\
&\wedge \forall_{d \in D \cap OT} \ \pi_{omap}(e)(d) \subseteq \cup \, sel(d)\}
\end{aligned}
$$

In materializing, we add content to the cells. In other words, we create an event log for the cells in the process cube view. In a $c \in cells$, each visible dimension is assigned to a value of that dimension, e.g., $c(activity)$=*create purchase order*, $c(timestamp) = T_{2020}$, and $c(item) = \{i_1, i_2, ..., i_{13502}\}$. We materialize the cells of the process cube with events. $events(c)$ are all the events corresponding to the first requirement (i.e., $\forall_{d \in D_{sel} \cap ATT} \ \pi_{vmap}(e)(d) \in c(d)$, $\forall_{d \in D_{sel} \cap OT} \ c(d) \subseteq \pi_{omap}(e)(d)$) which is different for object types and attributes. The second requirement (i.e., $\forall_{d \in D \cap ATT} \ \pi_{vmap}(e)(d) \in \cup \, sel(d)$, $\forall_{d \in D \cap OT} \ \pi_{omap}(e)(d) \subseteq \cup \, sel(d)$), which is also different for object types and attributes, makes sure that events are not filtered out. Events in the cell can be converted to an object-centric event log and used to apply process mining techniques.

Using earlier formalizations, we define process cube operations such as *slice*.

Definition 7 (Slice)**.** *Let $PCS = (D, val, gran)$ be a process cube structure and $PCV = (D_{sel}, sel)$ a view of PCS. For any $d \in D_{sel}$ and $V \in sel(d)$: $slice_{d,V}(PCV) = (D'_{sel}, sel')$ with $D'_{sel} = D_{sel} \backslash \{d\}$, $sel'(d) = \{V\}$ and $sel'(d') = sel(d')$ for $d' \in D \setminus \{d\}$.*

Through slicing a new cube view is produced and a dimension d is removed from the cube. As shown in Fig. 4, in slicing for the item dimension and value set $\{i_{13503}, i_{13504}, ..., i_{27005}\}$, the item dimension is removed and only events in which the item is a subset of $\{i_{13503}, i_{13504}, ..., i_{27005}\}$ remain in the cube view.

Definition 8 (Dice)**.** *Let $PCS = (D, val, gran)$ be a process cube structure and $PCV = (D_{sel}, sel)$ a view of PCS. Let $fil \in D_{sel} \nrightarrow U_h$ be a filter such that for any $d \in dom(fil)$: $fil(d) \subseteq sel(d)$. $dice_{fil}(PCV) = (D_{sel}, sel')$ with $sel'(d) = fil(d)$ for $d \in dom(fil)$ and $sel'(d) = sel(d)$ for $d \in D \setminus dom(fil)$.*

The difference between dice and slice is that through dicing, no dimension is removed and it limits the values for one or more dimensions. For example dicing is applicable based

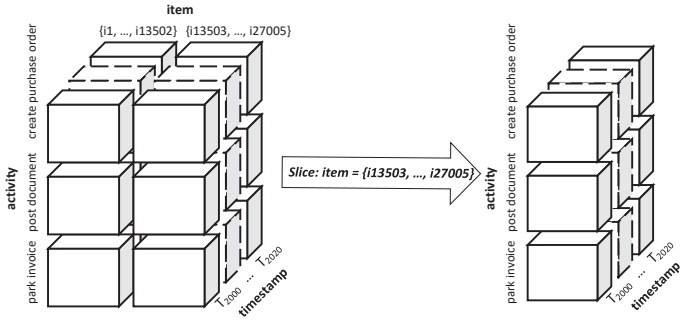


Fig. 4 The process cube view after slicing based on dimension item

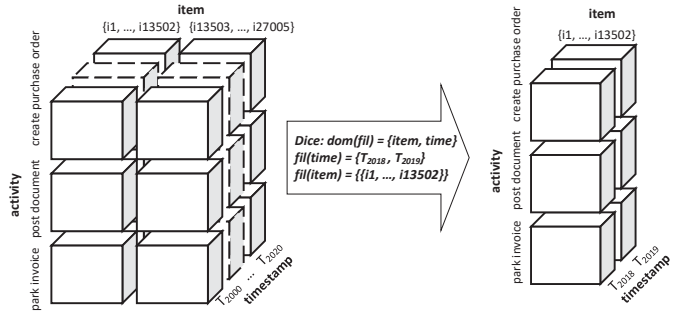


Fig. 5 The process cube view after dicing based on dimensions item and time

on two dimensions time and item, as shown in Fig. 5. In this example, we have new process cube view based on two filters: $fil(time) = \{T_{2018}, T_{2019}\}$ and $fil(item) = \{i_1, i_2, ..., i_{13502}\}$.

Definition 9 (Change Granularity)**.** *Let* $PCS = (D, val, gran)$ *be a process cube structure and* $PCV = (D_{sel}, sel)$ *a view of* PCS*. Let* $d \in D_{sel}$ *and* $G \in \mathbb{U}_h$ *such that:* $G \subseteq gran(d)$ *and* $\cup G = \cup sel(d)$*.* $chgr_{d,G}(PCV) = (D_{sel}, sel')$ *with* $sel'(d) = G$*, and* $sel'(d') = sel(d')$ *for* $d' \in D \setminus \{d\}$*.*

The dimensions in process cube view do not change during changing granularity, but the dimensions are shown in a more fine-grained or coarse-grained vision. In Fig. 3a the granularity for time dimension is year. However, in Fig. 3b, the granularity for time dimension is month. Having different levels of the granularity, we can compare processes in different levels of granularity.

V. Implementation

The approach has been implemented as a standalone Python application [1] by using PM4PY-MDL. The user can easily install the framework in Python 3.6. The procedure to create a cube in this framework is shown in Fig. 6. The functionalities of the framework are:

- It is possible to import object-centric event logs (The framework also accepts CSV and XES formats and automatically converts them to an object centric event log with one case notion).
- The created cube can be exported in a dump file and stored in the memory for future exploration.
- At any point in time, it is possible to export the object-centric event log as an object-centric event log or a traditional event log by selecting a case notion. It is also possible to show the process models of the selected cell(s).

[1] available in https://gitlab.com/Anahita-Farhang/process-cube

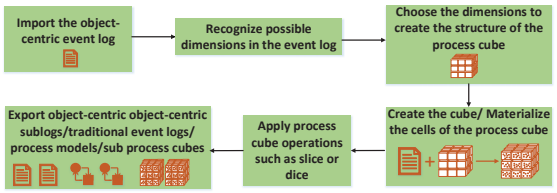


Fig. 6 An overview showing how the framework can be used. The input is an object-centric event log. By choosing the dimensions, the user can build the process cube and through process cube operations explore the cube. It is possible to have the output as an event log, a process model, and a sub cube

It is possible to explore the cube interactively through process cube operations such as slicing. The user can discover an MVP model, enhanced with performance/frequency information, and Object-centric Petri net for each cell. Fig. 7 shows a comparison between MVP models of a dice and the whole cube:

- In Box 1, it is possible to specify for each object type

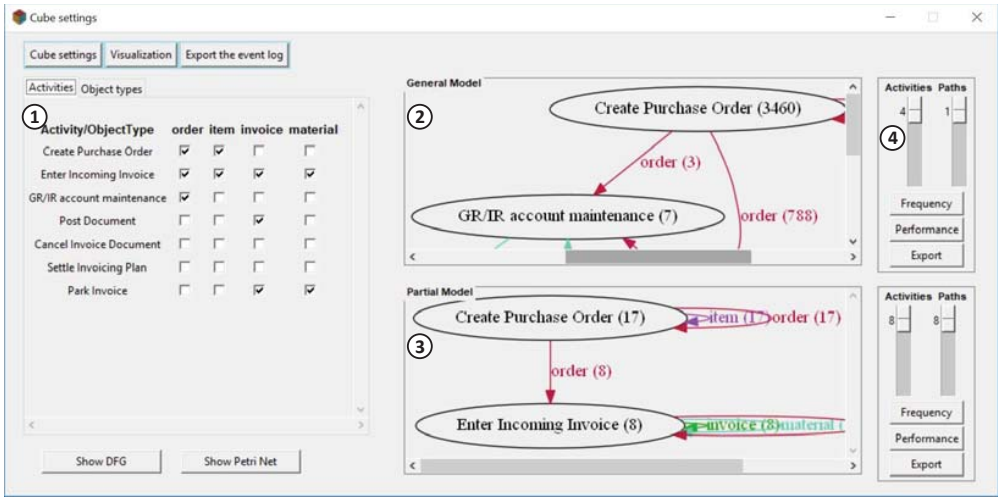


Fig. 7 Process Mining Cube: process comparison approach

the activities that are considered for that object type. For example, for the activity *create purchase order*, *orders* are involved, but *items* are not involved in the *post document*.

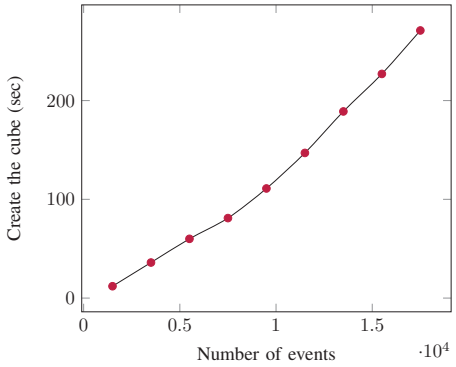
- The MVP model of the whole cube is shown in Box 2 by considering the filtering options in Box 1. MVP models are DFGs with colored arcs. In this figure, the color of the arcs related to the *order* is red.
- The MVP model of the specific *slice/dice* of the cube is represented in Box 3. Putting this model near the whole cube's model makes the comparison easier.
- In Box 4, the user can change the frequency of the nodes and edges appearing in the MVP model. There is a performance annotated version of MVP models that is reachable only by clicking on performance. It is possible to export the object-centric process model as an image in the desired address.

VI. Evaluation

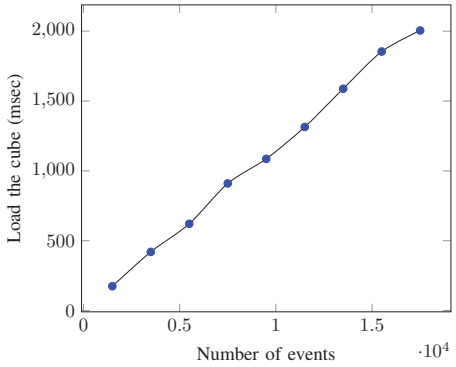
In this section, we analyze the scalability of the object-centric process cube tool. To assess the performance of our approach, we measure the scalability of the approach from two perspectives: creation time (e.g., the time required to create and materialize cells of the cube), and the loading time (e.g., the time required to import the cube). Results, shown in Fig. 8, have been done in three different settings: the time for creating/loading the cube in terms of the number of events in the event log (while keeping the number of object types and attributes constant), the number of object types in the event log (while keeping the number of events and attributes constant), and the number of attributes in the event log (while keeping the number of events and object types constant). Performance analysis of the cube with different settings shows the time required for creating/loading the cube increases linearly, non-linearly, and linearly when increasing number of events, object types, and attributes respectively. The proposed framework creates a process cube for an event log with 17500 events in almost 4 minutes.

VII. Conclusion

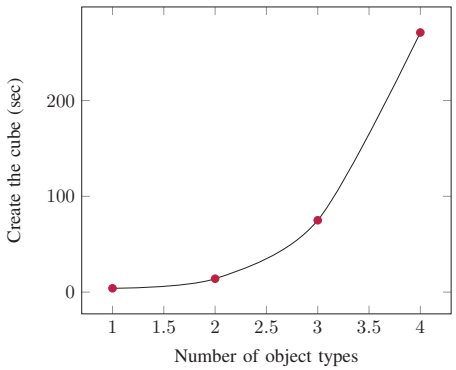
This paper bridges the gap between object-centric process mining and process comparison. Therefore, we proposed an object-centric process cube that organizes data through dimensions referring to case notions (i.e., process instances) and attributes in the event log. The proposed framework allows the users to explore the object-centric event logs interactively through the process cube operations such as slice. Furthermore, the proposed framework, which is publicly available, is able to discover object-centric process models from object-centric event logs extracted from the cells of the cube, which speeds up process comparison. For the future work, we aim to add more features related to performance to the object-centric process cube framework to facilitate the process analysis



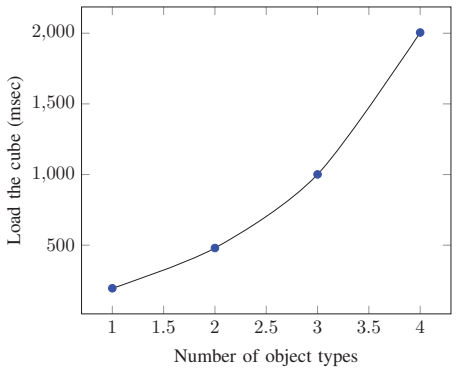
a.1) The difference in time required for creating the cube for different numbers of events: n_object types = 4, and n_attributes = 4



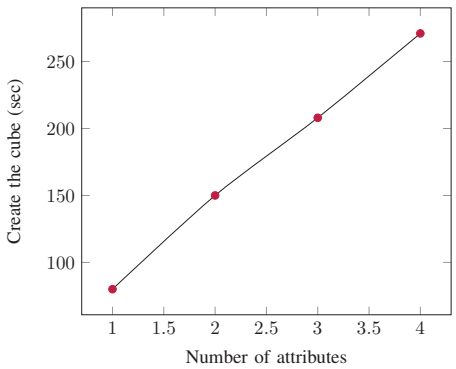
a.2) The difference in time required for loading the cube for different numbers of events: n_object types = 4, and n_attributes = 4



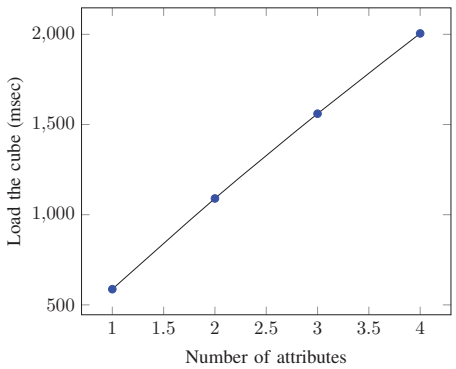
b.1) The difference in time required for creating the cube for different numbers of object types: n_events = 17500, and n_attributes = 4



b.2) The difference in time required for loading the cube for different numbers of object types: n_events = 17500, and n_attributes = 4



c.1) The difference in time required for creating the cube for different numbers of attributes: n_events= 17500 , and n_object types = 4.



c.2) The difference in time required for loading the cube for different numbers of attributes: n_events = 17500, and n_object types = 4.

Fig. 8 Performance analysis of creating (diagrams with red dots)/loading (diagrams with blue dots) the cube for the proposed approach based on a) the number of events b) the number of object types, and c) the number of attributes

ACKNOWLEDGMENTS

We thank the Alexander von Humboldt (AvH) Stiftung for supporting our research. Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy–EXC-2023 Internet of Production – 390621612.

REFERENCES

[1] A. Bolt and W. van der Aalst, "Multidimensional process mining using process cubes," in *Enterprise, Business-Process and Information Systems Modeling*. Springer, 2015, pp. 102–116.
[2] J. Ribeiro and A. Weijters, "Event cube: another perspective on business processes," in *International Conferences On the Move to Meaningful Internet Systems*. Springer, 2011, pp. 274–283.
[3] C. Chen, X. Yan, F. Zhu, J. Han, and S. Y. Philip, "Graph OLAP: a multi-dimensional framework for graph data analysis," *Knowledge and information systems*, vol. 21, no. 1, pp. 41–63, 2009.
[4] W. van der Aalst, "Object-centric process mining: Dealing with divergence and convergence in event data," in *International Conference on Software Engineering and Formal Methods*. Springer, 2019, pp. 3–25.
[5] T. Vogelgesang and H.-J. Appelrath, "Multidimensional process mining: a flexible analysis approach for health services research," in *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, 2013, pp. 17–22.
[6] W. van der Aalst, "Process cubes: Slicing, dicing, rolling up and drilling down event data for process mining," in *Asia-Pacific Conference on Business Process Management*. Springer, 2013, pp. 1–22.
[7] R. Andreswari and M. A. Rasyidi, "OLAP cube processing of production planning real-life event log: A case study," in *ICoIESE 2018*. Atlantis Press, 2019.
[8] M. R. H. Nik, W. van der Aalst, and M. F. Sani, "Bipm: Combining bi and process mining." in *DATA*, 2019, pp. 123–128.
[9] D. Cohn and R. Hull, "Business artifacts: A data-centric approach to modeling business operations and processes." *IEEE Data Eng. Bull.*, vol. 32, no. 3, pp. 3–9, 2009.
[10] K. Bhattacharya, C. Gerede, R. Hull, R. Liu, and J. Su, "Towards formal analysis of artifact-centric business process models," in *International Conference on Business Process Management*. Springer, 2007, pp. 288–304.
[11] X. Lu, M. Nagelkerke, D. van de Wiel, and D. Fahland, "Discovering interacting artifacts from ERP systems (extended version)," *BPM reports*, vol. 1508, 2015.
[12] G. Li, R. M. de Carvalho, and W. van der Aalst, "Automatic discovery of object-centric behavioral constraint models," in *International Conference on Business Information Systems*. Springer, 2017, pp. 43–58.
[13] A. Berti and W. van der Aalst, "Extracting multiple viewpoint models from relational databases," in *Data-Driven Process Discovery and Analysis*. Springer, 2018, pp. 24–51.
[14] W. van der Aalst and A. Berti, "Discovering Object-centric Petri nets," in *Fundamenta Informaticae*, 2020.
[15] M. Gupta and A. Sureka, "Process cube for software defect resolution," in *Asia-Pacific Software Engineering Conference*, vol. 1. IEEE, 2014, pp. 239–246.