Preserving complex object-centric graph structures to improve machine learning tasks in process mining

Jan Niklas Adams [a,*], Gyunam Park [a], Wil M.P. van der Aalst [a,b,c]

[a] Chair of Process and Data Science, RWTH Aachen University, Ahornstraße 55, Aachen, 52074, North Rhine-Westphalia, Germany
[b] Fraunhofer FIT, Schloß Birlinghoven, Konrad-Adenauer-Straße, Sankt Augustin, 53757, North Rhine-Westphalia, Germany
[c] Celonis SE, Theresienstr. 6, Munich, 80333, Bavaria, Germany

A B S T R A C T

Interactions of multiple processes and different objects can be captured using object-centric event data. Object-centric event data represent process executions as event graphs of interacting objects. When applying machine learning techniques to object-centric event data, the event log has to be flattened into sequential process executions used as input for traditional process mining approaches. However, sequentializing the events by flattening removes the graph structure of object-centric event data and, therefore, constitutes an information loss. In this paper, we present a general approach to preserve the graph structures of object-centric event data across machine learning tasks in process mining. We provide two different techniques to preserve these structures depending on the required input format of machine learning techniques: as direct graph encodings or as graph embeddings. We evaluate our contributions by applying three different predictive process monitoring tasks to direct graph encodings, graph embeddings, and flattened event logs. Based on the relative performances, we assess the information contained in the graph structure of object-centric event logs that is, currently, lost in machine learning approaches for traditional process mining. Furthermore, we compare different graph embedding techniques for object-centric event logs to derive recommendations for future research and deployment. To conclude, we assess the improvement that graph embeddings constitute over state-of-the-art approaches that capture object information through specific features. Our contributions improve predictive process monitoring on object-centric event data and quantify the potential performance increases of predictive models. These contributions are especially relevant when looking at real-life applications of predictive process monitoring which are often applied to information systems with relational databases that contain multiple objects and, currently, still enforce flattening.

1. Introduction

Processes are omnipresent in today's world. Most business activities are conducted with the (implicit) notion of a process, ranging from manufacturing (ElMaraghy et al., 2009) to business workflows (Georgakopoulos et al., 1995). These processes are generally supported by information systems (Dumas et al., 2018). The execution of processes leaves traces of data in information systems. These data can be extracted to analyze the underlying process (Calvanese et al., 2015). Techniques delivering insights and operational support concerning the underlying process are summarized under the term *process mining* (van der Aalst, 2016). Process mining techniques are usually grouped in four categories: Process discovery (Augusto et al., 2019), conformance checking (Carmona et al., 2018), process enhancement (de Leoni, 2022), and operational support through the application of machine learning techniques such as predictive process monitoring (Francescomarino and Ghidini, 2022) or clustering (Zandkarimi et al., 2020).

Input data for process mining techniques come in form of an *event log*. An event log consists of a set of *events*. Each event is associated with an *activity*, a *timestamp*, a *case identifier*, and additional data attributes. All events belonging to the same case identifier are grouped to an event sequence referred to as *case*. Current process mining techniques assume such an event log as input. In many cases, this is a valid assumption, as information systems like business process management systems or case management/ticketing systems work on a single case notion for which we can record a sequence of events (Weerdt and Wynn, 2022).

However, the vast majority of processes is more complex. Events are not just related to a single case identifier but to multiple objects simultaneously. In reality, the support of most processes requires information systems with complex relational databases (de Murillas

* Corresponding author.
E-mail addresses: niklas.adams@pads.rwth-aachen.de (J.N. Adams), gnpark@pads.rwth-aachen.de (G. Park), wvdaalst@pads.rwth-aachen.de (W.M.P. van der Aalst).

et al., 2020; Jans and Soffer, 2017), expressing the interactions of different objects (entities) that events are related to Fahland (2022). Prominent examples of such information systems are like Enterprise Resource Planning (ERP) systems (Ingvaldsen and Gulla, 2007) or Manufacturing Executions Systems (MES) (Yang et al., 2014). Such complex process executions are not simple sequences of events, but complex interactions of different objects and their individual paths through a process. This complex behavior can be captured using event graphs that are constructed by merging the events of different objects at multiple interaction points (Adams et al., 2022c; Esser and Fahland, 2021; Waibel et al., 2022).

The complex graph structures of event data need to be reduced to a sequential event log format to apply traditional process mining techniques. This process is called *flattening* (Calvanese et al., 2015). Since flattening forces the complex structure of object interactions into a relatively less expressive sequential structure, not all the structural information about objects and their interactions can be preserved. The quality problems of flattening are known in process mining: Flattening leads to disappearing events (deficiency), duplicated events (convergence) and indistinguishable event-object relationships with wrong precedence constraints (divergence) (van der Aalst, 2019; Weber et al., 2015; Waibel et al., 2022).

To answer the shortcoming of traditional event data when capturing event data related to multiple objects, *object-centric event logs* have been proposed (van der Aalst and Berti, 2020). In an object-centric event log, an event is no longer associated with exactly one case identifier but with multiple objects. Therefore, object-centric event logs are able to capture interactions between objects and allow the individual tracking of objects through the process. Recently, the move from the traditional sequential process execution concept to a graph-based process execution concept has been proposed (Adams et al., 2022c). Building and adapting process mining techniques on top of the graph-based process execution concept prevents the flattening of the event log, preserving the structural information of the object-centric event log.

Machine learning techniques are actively utilized in process mining to provide operational support. *Predictive Process Monitoring* (PPM) (Francescomarino and Ghidini, 2022) summarizes all techniques to predict the future of ongoing process executions. A wide range of prediction targets are considered, from process execution's outcomes (Teinemaa et al., 2019), numerical measures such as performance indicators (Castellanos et al., 2006), or future control-flow behavior such as the next activity (Tax et al., 2017). Clustering of process executions provides insights into similar process executions, pointing to potential problems or outcomes (Zandkarimi et al., 2020). In general, all of these techniques start from the event log, extract features (de Leoni et al., 2016), and encode them for the corresponding machine learning technique. Applying all of these techniques to more complex object-centric event data with multiple objects per event requires the flattening of the event log, as proposed by current state-of-the art techniques (Galanti et al., 2023). However, as already discussed, flattening is associated with quality problems ranging from removed information over duplicated entries to incorrect precedence constraints. This raises two main questions:

RQ1 How much information is lost through the process of flattening and removing the graph structure of process executions?

RQ2 What is the best way to preserve the graph structure of object-centric event data for machine learning applications in process mining?

In this paper, we tackle these two research questions by providing a general machine-learning approach to preserve the graph structure of object-centric event data to improve machine learning results. Our contributions are as follows:

C1 We define a general approach for preserving the graph structure of object-centric event data for machine learning applications using graphs and graph embeddings.

C2 We quantify the potential value of preserving the graph structure by isolating its predictive value for different predictive process monitoring tasks.

C3 We compare our proposed technique of graph embeddings against current state-of-the-art methods that capture object-centric structures through manually designed features. To this end, we quantify the predictive delta between our proposed method and these manual graph embeddings.

C4 We provide implementations and our experiments in publicly available repositories which can be used for new developments of machine learning applications and adaption of current applications.

The remainder of this paper is structured as follows. We introduce the related work on object-centric process mining and machine learning in process mining in Section 2. The preliminaries on object-centric event data are presented in Section 3. We present our framework to preserve the structure of object-centric event data for machine learning tasks and present examples of structural information loss through flattening in Section 4. In Section 5, we evaluate our framework by quantifying the information contained in the graph structure of object-centric process executions. Furthermore, we compare our proposed method of graph embeddings to state-of-the-art techniques of embedding information about objects. We conclude this paper in Section 6.

2. Machine learning and object-centric process mining

The object-centric nature of processes has been known for a long time (Cohn and Hull, 2009). More than a decade ago, the problem of algorithms to handle object-centric event data was formulated as one of the main challenges in process mining (van der Aalst et al., 2011). In this section, we will first discuss existing work on object-centric processes from a modeling, data storing, and algorithmic perspective and discuss the recently introduced graph-based view on object-centric event data. Furthermore, we will introduce graphs in machine learning and depict the shortcoming of machine learning techniques in process mining to incorporate the graph structure of object-centric event data.

2.1. Processes with multiple case notions

One of the earliest modeling techniques for interacting processes were artifact-centric business process models (Calvanese et al., 2014; Meyer et al., 2013; Popova et al., 2015) and Proclets (Fahland, 2019). Both modeling notations show a collection of connected, individual models per object type rather than one single model capturing explicit causality constraints. Object-centric behavioral constraint (OCBC) models (van der Aalst et al., 2017) explicitly capture causality constraints between object types and depict them in one single model. Since the discovery of OCBC models is not scalable, object-centric Petri nets (van der Aalst and Berti, 2020) were proposed, unifying all object types in one comprehensive model and exhibiting promising scalability for process discovery. Other object-centric modeling notations include Information Systems Modeling Suite (ISML) (van der Werf and Polyvyanyy, 2020), Catalog-nets (Ghilardi et al., 2020), and DB-nets (Montali and Rivkin, 2017)

To enable process mining, one must extract and store data in the appropriate format for process discovery algorithms and analysis of object-centric processes. Different event log formats were proposed. Artifact-centric event logs (Nooijen et al., 2012; Moctar-M'Baba et al., 2022) are the earliest storage format for event data with multiple objects per event. eXtensible object-centric (XOC) event logs (Li et al., 2018) event logs were proposed to be used for the discovery and analysis of OCBC models, storing the whole evolution of the underlying database. This has obvious scalability issues for large event logs. Object-centric event logs (OCELs) (Ghahfarokhi et al., 2021) were recently proposed as a lightweight way to store object-centric event data,

recording the objects per event and their attributes. Furthermore, graph databases (Esser and Fahland, 2021) have recently been introduced as a storage format for object-centric event data.

Based on an event log capturing real-life process executions, process mining algorithms provide insights into the executed process. Several specific algorithms have been developed for different storage formats and different modeling notations of object-centric processes. For artifact-centric processes, discovery algorithms (Nooijen et al., 2012; Lu et al., 2015) and conformance checking techniques (Fahland et al., 2011) have been introduced. Process models can, also, be discovered from graph databases (Eldin et al., 2022). Furthermore, OCBC models can automatically be discovered from XOC event logs (Li et al., 2017). However, the most variety of algorithms exists for object-centric event logs and object-centric Petri nets. Next to object-centric variants (Adams et al., 2022c), two different types of models can be discovered from OCELs: multiple viewpoint models (Berti and van der Aalst, 2019) and object-centric Petri nets (van der Aalst and Berti, 2020). Object types exhibiting similar control-flow behavior can be clustered using a Markov Directly-Follows Multigraph (Jalali, 2022). Furthermore, conformance-checking techniques for OCELs and object-centric Petri nets were recently proposed (Adams and van der Aalst, 2021; Park and van der Aalst, 2022). Model-based performance analysis with novel performance measures was proposed as a data-driven way to analyze the interplay of processes (Park et al., 2022). At last, a native feature extraction and encoding for OCELs was proposed, enabling PPM for object-centric event logs (Adams et al., 2022a).

We built our method and evaluation based on existing work on object-centric event logs. We chose object-centric event logs for the following reason: A rich ecosystem of algorithms and implementation around object-centric event logs enables us to integrate our method and make it available for future research. We implement our method in the recently introduced OCPA library (Adams et al., 2022b). However, our method can equally be applied to other existing storage formats such as graph databases or XOC event logs. All storage formats are united by the graph-based view on event data, in contrast to the sequence-based view traditionally taken in process mining.

2.2. Graphs in machine learning

Many real-life settings closely resemble a graph structure and require adapted machine learning techniques working on graph input data. These application domains include traffic analysis (Zhang et al., 2022), visual scene understanding (Ding et al., 2022), or chemistry and molecules (Ma et al., 2022). A plethora of machine learning tasks has been adapted to the graph input setting, e.g., prediction (Ali et al., 2022), outlier detection (Li et al., 2022), or clustering (Mishra et al., 2022). Tasks can generally be performed on the graph level (e.g., classifying properties of a molecule), the edge level (e.g., predicting the traffic delay on a certain street), or the node level (e.g., classifying outliers in a network).

Graph isomorphism is a fundamental problem in analyzing graphs. The problem can be solved in quasipolynomial time (Babai, 2016). In machine learning applications, however, instead of determining whether two graphs are identical, one aims to analyze the similarity between graphs, e.g., classifying graphs based on their similarity. This can be done by learning a model directly on the graph data, i.e., a graph neural network (Kipf and Welling, 2017; Wu et al., 2021). Throughout multiple convolutional layers, values are propagated along the edges of the graph according to learned weights. The final layers can then be pooled to perform a classification/regression. Another approach to computing the similarity between graphs is to learn a representation of the graph as a vector, called graph embedding, and adopt machine learning models to learn the similarity.

Many prior studies have explored learning graph representations. The methods for graph embedding can be grouped into three categories: *graph kernel-based methods*, *deep learning-based methods*, and *representation-based methods*. First, graph kernel-based methods decompose a graph into small sub-structures and build graph kernels using the similarity defined over these components. Different graph kernels decompose graphs in different manners. *Graphlet* kernels (Yanardag and Vishwanathan, 2015) are based on subgraphs up to a fixed size, whereas *Weisfeiler–Lehman* graph kernels (Shervashidze et al., 2011) are based on subtree patterns. *Shortest path* kernel (Borgwardt and Kriegel, 2005) are based on the shortest paths between node pairs.

Deep learning-based graph embedding applies deep learning models on graphs. *Graph2Vec* (Narayanan et al., 2017) extract rooted subgraph features using the Weisfeiler–Lehman kernel and learn graph embedding by passing them to a *Doc2Vec* (Mikolov et al., 2013) model. *GL2Vec* (Chen and Koga, 2019) extends *Graph2Vec* by using line graphs to incorporate edge features into graph representations.

Representation-based graph embedding uses statistical properties of a graph to generate a graph signature vector to represent the graph. Such statistical properties include local characteristics such as node's and its neighbors' degrees (Cai and Wang, 2018). More advanced methods use global features such as *skew spectrum* (Kondor and Borgwardt, 2008) and its successor, *graphlet spectrum* (Kondor et al., 2009).

2.3. Machine learning in process mining

This section discusses the central methodological gap our paper addresses: On the one hand, research interest in object-centric event data and its graph-based structure has drastically increased over the last decade. On the other hand, an extensive amount of general machine learning methods based on graphs has been proposed. However, the graph-based structure of object-centric event data has, so far, been neglected in machine learning tasks in process mining.

Machine learning has become a central pillar in providing operational support for processes. While predictive process monitoring (Francescomarino and Ghidini, 2022) has received the most attention, other machine learning tasks such as clustering (Zandkarimi et al., 2020), outlier detection (Sani et al., 2018), sampling (Bauer et al., 2019), or concept drift adaptation (Chamorro et al., 2022) have also been adapted to process mining use cases.

Traditional process mining is built around the assumption of a traditional event log, i.e., an event log where each event is associated with exactly one object of the same type, forming an event sequence. Applying any machine learning technique for process mining follows a rigid pipeline: First, the process data recorded in information systems is transformed into traditional event log format Calvanese et al. (2015) and, second, the traditional event log is transformed into the required input format for the machine learning technique. In this two-step process, research progress focused on advancing the second step of this pipeline, i.e., extracting features and transforming the traditional event log into different input formats to support all kinds of machine learning techniques: based on tabular data (de Leoni et al., 2016), sequential data (Tax et al., 2017; Leontjeva et al., 2015; Evermann et al., 2017), and graphs constructed from instance graphs (Chiorrini et al., 2021). While this has led to important developments of, especially, predictive process monitoring, the first part of this pipeline has stayed mostly untouched, leaving opportunities for further improvement. *This paper addresses how graph-based process executions from object-centric event logs can be preserved throughout this two-step pipeline.*

Incorporating object-centric event logs into machine learning tasks such as predictive process monitoring has recently been studied by some papers. However, the proposed approaches still require the flattening of the event log, removing structural information. Galanti et al. proposed a framework for predictive analytics on object-centric event logs (Galanti et al., 2023) that incorporates object interactions as additional features into a traditional predictive process analytics framework, namely aggregations over object attributes, the number of objects of a certain type per event, and the percentage of objects that have run through an activity. *These features should capture the object flow and*

object associations of events, essentially functioning as some manual embedding. They report an increase in predictive performance using these three additional object-centric features. However, as the event log still needs to be flattened and the added features are only a small selection of possible features, these features cannot compensate for the structural information loss in flattening (cf. extensive examples in Section 4: Fig. 3 (rework cannot be distinguished using these features), Fig. 4 (individual object flows cannot be substituted using these features), and Fig. 5(activities cannot be associated to a specific object using these features)). Adams et al. (2022a) introduced a comprehensive framework for object-centric feature extraction and encoding forming an object-centric adaptation of de Leoni et al. (2016), proposing object-centric adaptations of commonly used features, e.g., moving from one preceding activity in traditional process mining to multiple preceding activities in object-centric feature extraction or using object-centric performance metrics. The features building on the object perspective can also be used as a manual embedding of object information. While this framework provides an extensive adaptation and implementation of new features, these features can either be encoded directly as a graph and used in, e.g., a graph neural network, or the structure can be flattened to fit tabular/sequential machine learning models. Therefore, the graph-based structure of object-centric process executions cannot be preserved for machine learning techniques with vector input using the authors' framework. In this work, we close this gap by using graph embeddings to preserve the structure of process executions for machine learning tasks with vector input. Furthermore, we also quantify the information contained in the graph-based process executions through a comparison of predictive utility between object-centric structures and flat event data structures. We compare the amount of predictive information captured by our proposed graph embeddings to the manual embedding approaches contained in Galanti et al.'s (2023) and Adams et al.'s (2022a) work.

3. Object-centric event data

This section introduces key concepts for object-centric event data. We accompany these with examples of an object-centric event log and object interactions.

First, we introduce some notations used throughout this paper. An event log consists of events. The universe of events is denoted as $\mathcal{E}$. Each event happens at a certain point in time where $\mathcal{T}$ denotes the universe of timestamps. Event logs record events that are associated with objects. The universe of objects is denoted with $\mathcal{O}$. Each object is of exactly one type, describing an instance of this type. The universe of object types is denoted by $\mathcal{OT}$. The object type of an object is mapped through $\pi_{type} : \mathcal{O} \rightarrow \mathcal{OT}$. Each event describes the occurrence of an activity. The universe of activities is given by $\mathcal{A}$. Events are, furthermore, associated with different attributes and values to these attributes. The universe of attributes is given by $\mathcal{AT}$, and the universe of attribute values is denoted as $\mathcal{V}$.

A sequence $\delta : \{1, \dots, n\} \rightarrow X$ assigns an order to elements of a set X and is denoted with $\langle x_1, \dots, x_n \rangle$. The subsequences of a sequences are defined by $subseq(\langle x_1, \dots, x_n \rangle) = \{\langle x_1 \rangle, \dots, \langle x_n \rangle, \langle x_1, x_2 \rangle, \dots, \langle x_{n-1}, x_n \rangle, \dots, \langle x_1, \dots, x_{n-1} \rangle, \langle x_2, \dots, x_n \rangle, \langle x_1, \dots, x_n \rangle\}$.

Definition 1 (*Object-Centric Event Log*)**.** An object-centric event log is a tuple $L = (E, OT, O, \pi_{time}, \pi_{trace}, \pi_{obj}, \pi_{act}, \pi_{att})$ consisting of

1. a set of events $E \subseteq \mathcal{E}$,
2. a set of object types $OT \subseteq \mathcal{OT}$ and objects $O \subseteq \mathcal{O}$,
3. a function $\pi_{time} : E \rightarrow \mathcal{T}$ mapping each event to its timestamp,
4. a function $\pi_{trace} : O \rightarrow E^*$ mapping each object to a sequence of events such that $\forall_{o \in O}\ \pi_{trace}(o) = \langle e_1, \dots, e_n \rangle \wedge \forall_{i \in \{1,\dots,n-1\}}\ \pi_{time}(e_i) \leq \pi_{time}(e_{i+1})$,
5. a function providing the objects associated to an event $\pi_{obj}(e) = \{o \in O \mid e \in \pi_{trace}(o)\}$,
6. a function $\pi_{act} : E \rightarrow \mathcal{A}$ mapping events to activities,
7. a function $\pi_{att} : E \times \mathcal{AT} \nrightarrow \mathcal{V}$ mapping events and attributes to attribute values.

An object-centric event log consists of events and objects. Events are related to objects through the π_{trace} function, which maps each object to a sequence of events. Events can belong to the event sequences of multiple objects of different types, expressing the interaction between objects and object types. The left-hand side of Fig. 2 depicts an example of an object-centric event log. An event (a row in the table) can be associated with objects of three types: job offers, applications, and interviews. Next to the activities and the objects of different types, each event is also associated with a timestamp. Each object is associated with a sequence of events, e.g., $\pi_{trace}(i1) = \langle e_3, e_9, e_{10} \rangle$ is the sequence of events associated with interview *i1*.

Object relationships are recorded through co-appearances at events. If an event refers to two different objects, these objects are in some relationships. We can build the graph of object relationships by traversing the event log for co-appearing objects (Adams et al., 2022c).

Definition 2 (*Object Graph Adams et al., 2022c*)**.** Let $L = (E, OT, O, \pi_{time}, \pi_{trace}, \pi_{obj}, \pi_{act}, \pi_{att})$ be an object-centric event log. All object relationships of the event log are captured in the object graph $OG_L = (O, I)$ with $I = \{\{o, o'\} \mid o \neq o' \wedge \exists_{e \in E}\ \{o, o'\} \subseteq \pi_{obj}(e)\}$.

The right-hand side of Fig. 2 depicts the object graph derived from the object-centric event log. The nodes are the objects appearing in the event log. Edges are introduced if two objects share an event. Using the object graph, one can spot dependencies between objects. For example, job offer *o1* and interview *i1* do not share events, however, they are transitively connected through application *a2*.

4. Encoding structural information of object-centric event data

We present our method for preserving complex object-centric graph structures in this section. We accompany our method with in-depth examples of structural information loss when using state-of-the-art techniques, i.e., flattening. Our framework for encoding structural information of the object-centric for machine learning tasks is depicted in Fig. 1 and highlighted in orange. Starting from the object-centric event log, we extract the graph-based process execution. Subsequently, we differentiate between two types of machine learning tasks depending on their required input format: Graph-based input format and vector input format. Machine learning tasks requiring graph-based input format can directly be applied on the process executions with trivial modification. Machine learning tasks that work on vector input – arguably the most frequently assumed input format – can, however, not directly be applied to graph-based process executions. Graphs need to be embedded into a vector (Cai et al., 2018).

4.1. Extracting graph-based process executions

Each object is associated with a sequence of events in an object-centric event log. Objects that share events influence each other. Merging the event sequences of two connected objects results in an event graph. We generalize this to an object-centric process execution definition of event graphs spanned by multiple, connected objects.

Definition 3 (*Process Execution Adams et al., 2022c*)**.** Let $L = (E, OT, O, \pi_{time}, \pi_{trace}, \pi_{obj}, \pi_{act}, \pi_{att})$ be an object-centric event log. For a set of objects $X \subseteq O$ forming a connected subgraph in OG_L, a process execution is an event graph $p_X = (E_X, D_X)$ composed of

1. nodes $E_X = \{e \in E \mid \pi_{obj}(e) \cap X \neq \emptyset\}$, and
2. edges $D_X = \{(e, e') \in E_X \times E_X \mid \exists_{o \in X}\ \langle e, e' \rangle \in subseq(\pi_{trace}(o))\}$.

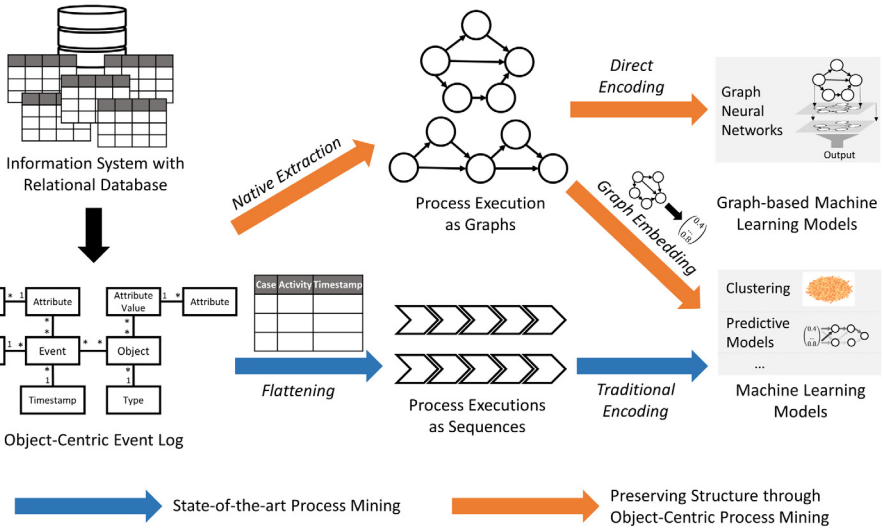

Fig. 1. Overview of our approach to preserving object-centric graph structures for machine learning tasks in process mining. In traditional process mining, object-centric event data are flattened first; the structure of the event log containing objects, frequencies, and interactions is lost. We either use the graph-based event structure directly or transform into a vector representation using graph embeddings.

Object-Centric Event Log

ID	Activity	Job Offer	Application	Interview	Timestamp
e_1	Submit Draft	o1			15.06.
e_2	Accept Draft	o1			18.06.
e_3	Set Interview Date			i1	20.06.
e_4	Publish Opening	o1			25.06.
e_5	Receive Application	o1	a1		26.06.
e_6	Receive Application	o1	a2		29.06.
e_7	Receive Application	o1	a3		30.06.
e_8	Reject Applications		a1, a3		05.07.
e_9	Invite to Interview		a2	i1	06.07.
e_{10}	Change Interview Date			i1	07.07.
e_{11}	Offer Postition	o1	a2		20.07.
e_{12}	Accept Position	o1	a2		25.07.

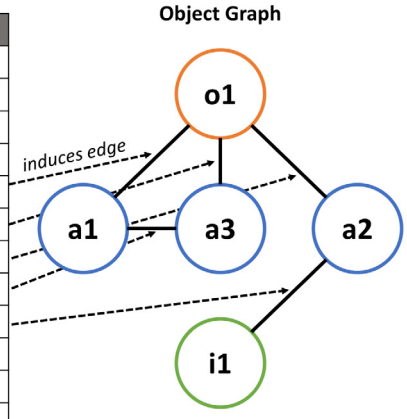

Fig. 2. Example of an object-centric event log and the object graph that is induced by the event log.

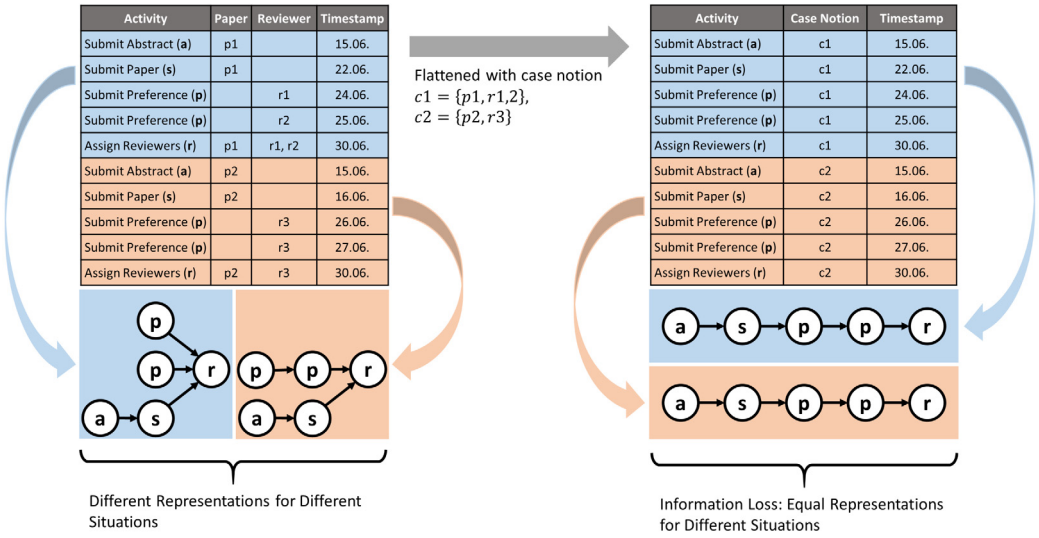

Fig. 3. When flattening and using the event activity attribute as feature, one cannot distinguish between two situations where one activity is either rework for one object or associated with a different object.

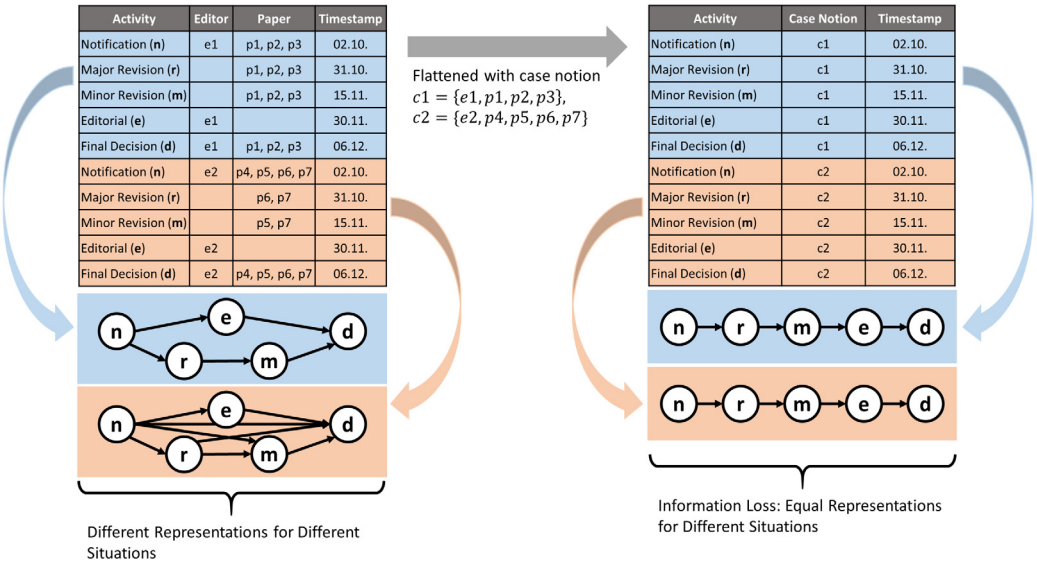

Fig. 4. The two process executions contain different numbers of objects and different individual object flows. However, they become indistinguishable when flattening.

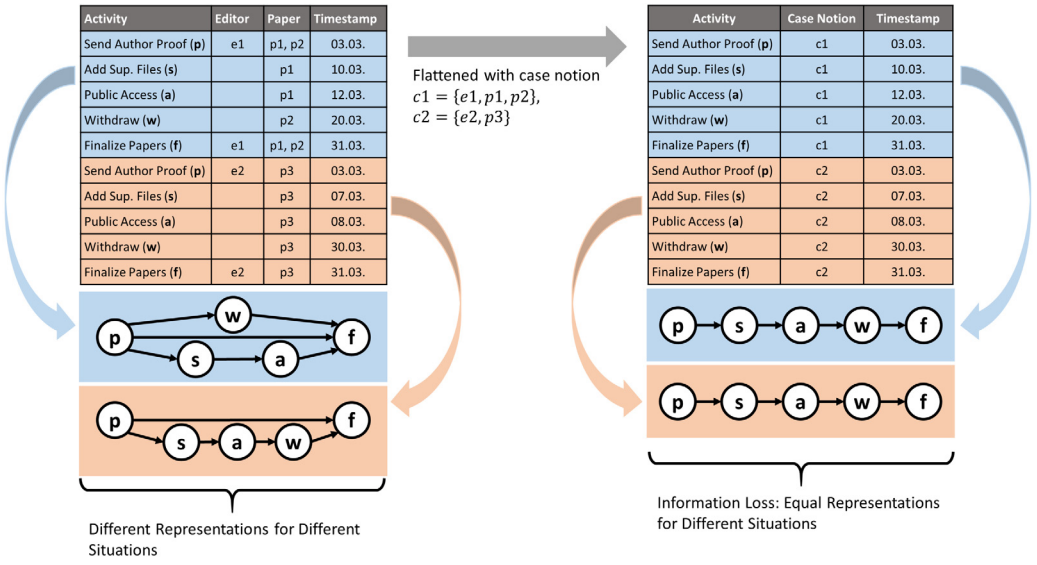

Fig. 5. The two process executions show two situations where an activity is associated with different objects. This information is lost when flattening.

A process execution contains the precedence constraints of events induced by different, connected objects in the form of a graph. Process executions capturing all the object relationships in an event log can be extracted by computing the connected components of the object graph. Among other extraction techniques (Adams et al., 2022c), we consider only an extraction through connected components for our definitions.

Definition 4 (*Execution Extraction*). Let $L=(E, OT, O, \pi_{time}, \pi_{trace}, \pi_{obj}, \pi_{act}, \pi_{att})$ be an object-centric event log. $pex(L) = \{p_X \mid X \subseteq O \wedge X \text{ is a connected component in } OG_L\}$ are the process executions extracted using the connected components of the object graph.

We can transform a graph-based process execution to the sequential case concept traditionally used in process mining to assess the impact of flattening. We do this by sorting events according to execution time and forcing them into sequential order. Note that, we could also flatten the process execution using only the events of a single object type (van der Aalst, 2019). However, flattening to a single object type is almost always associated with problems of disappearing events or duplicated events. Using all connected objects and flattening all associated events (Calvanese et al., 2015), we avoid these problems and preserve all events without changing event frequencies. Therefore, we choose the second form of flattening as a baseline.

Definition 5 (*Flattening*). Let $L = (E, OT, O, \pi_{time}, \pi_{trace}, \pi_{obj}, \pi_{act}, \pi_{att})$ be an object-centric event log. A process execution $p_X = (E_X, D_X)$ can be flattened into an event sequence through $flattening((E_X, D_X)) = (E_X, \{(e, e') \in E_X \times E_X \mid \pi_{time}(e) \leq \pi_{time}(e') \wedge \neg\exists_{e'' \in E_X} \pi_{time}(e) \leq \pi_{time}(e'') \leq \pi_{time}(e')\})$. The whole event log can be flattened by flattening each process execution $flat(L) = \{flattening(p_X) \mid p_X \in pex(L)\}$.

We provide three different end-to-end applications of our definitions in Figs. 3, 4, and 5. These depictions show three situations where flattening leads to an information loss. On the left-hand side, we show the object-centric event log and the corresponding process executions labeled with the event activities. On the right-hand side, we show the flattened (traditional) event log and the corresponding sequential traditional cases. In all three cases, situations that are distinguishable using an object-centric event log and graph-based process executions become indistinguishable when flattening. Moreover, when considering

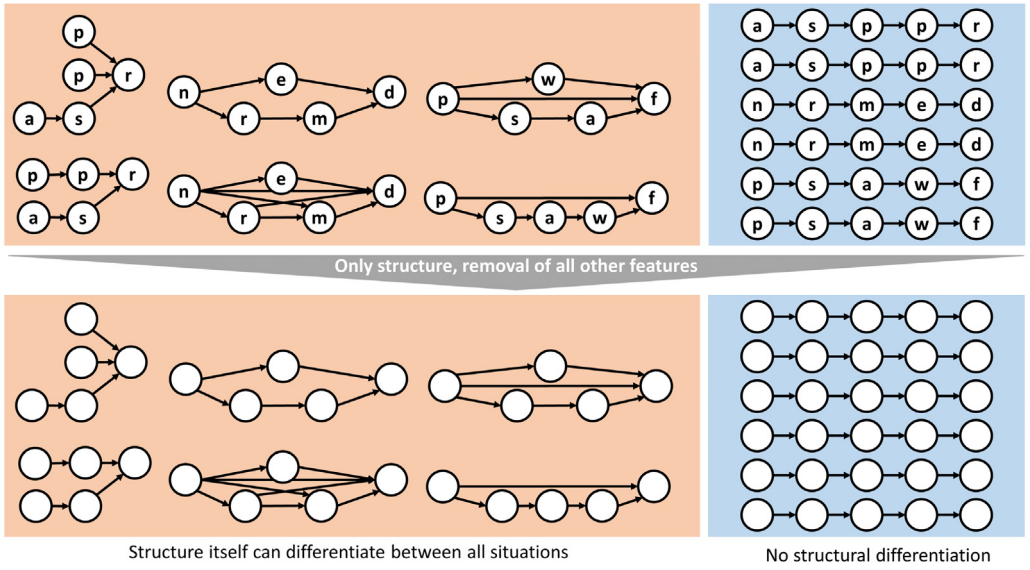

Fig. 6. Features such as the event's activity can substitute or enrich graph structure contained in the event log. When looking at only the structure of our examples without any feature, all six situations become indistinguishable when flattening but are perfectly distinguishable when using graph-based process executions.

only the structure without additional features such as the event activity, flattening makes all situations indistinguishable while all situations are distinguishable when using graph-based process executions (cf. Fig. 6). Please note that these situations might become distinguishable after flattening when using features other than the event's activity. However, for these features, one could equally well construct examples where they fail to distinguish other situations. If there is a (set of) feature(s) that can perfectly distinguish situations, this would constitute a perfect graph embedding for process executions.

In the following, we discuss the three examples in more depth. The example in Fig. 3 depicts a situation in which the rework of an activity and the presence of two objects for which the same activity is executed become indistinguishable through flattening. This stems form a general problem associated to flattening: When forcing all events into sequential format, it becomes untraceable what the object origin of these events is. Activity repetitions and additional objects with activity executions become indistinguishable.

The example in Fig. 4 depicts a situation where flattening hides the number of objects and their individual object traces. While the second process execution shows a great variance of individual object flows — some skipping major revisions, some going directly to the final decision, some skipping minor revision — the first process execution shows an equal flow with less objects. These differentiations are lost when flattening as it is impossible to trace objects' event sequences when there is only one sequence for all objects.

The example in Fig. 5 depicts a situation in which the object association of an (arguably important) activity gets lost when flattening. The first process execution contains two papers of which one is withdrawn and the other one has supplementary files added and public access activated. The second process execution contains only one paper where supplementary files are added, public access is activated but the paper is withdrawn afterwards. These situations become indistinguishable in the flattened event data.

The preceding examples are built on process executions labeled with the event activity feature. Event features can either enrich or substitute structural information. We aim to investigate the value of the structure that is lost when flattening. Therefore, we isolate the effect of only the structure by not enriching it with any features for the remainder of this paper. Fig. 6 depicts the consideration of only the structure and the information carried by it: All six situations are distinguishable when considering the structure of object-centric process executions but become completely indistinguishable when considering traditional, sequential cases.

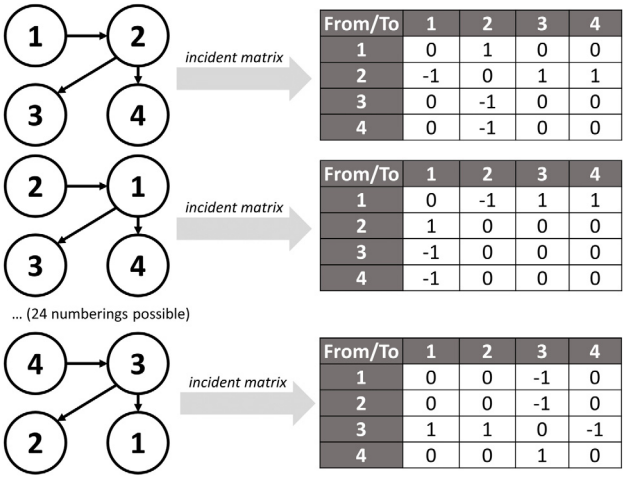

Fig. 7. There is no unique way to transform a graph into a vector representation. Depending on the enumeration of the exemplary graph's nodes, one can construct an incidence matrix in 24 different ways.

4.2. Machine learning on graph-based process executions

To highlight the necessity for graph embeddings, we will first demonstrate the problems of transforming graphs into vectors. We depict a small graph in Fig. 7. When translating a graph to a vector, one needs to find a function that will yield the same vector for the same graph. Simple strategies such as constructing the incident matrix do not work as there are different results for the same graph depending on the assignment of row/column indices to nodes. This is depicted in Fig. 7. We retrieve 24 different incident matrices for the same graph. Using these as vector embeddings, a machine-learning model would assume that these are 24 different graphs.

Graph embeddings address the problem of translating similar graphs to similar vector representations as accurately as possible. While doing so, it aims to represent the graph as low dimensional vectors so that machine learning models can extract useful information.

Definition 6 (*Graph Embedding*)**.** Let $G = (V, E)$ be a graph with nodes V and edges E. A graph embedding $embed(G) \in \mathbb{R}^n$ maps a graph to a vector of length $n \in \mathbb{N}$.

As introduced in Section 2.2, graph embedding methods are categorized into *graph kernel-based methods*, *deep learning-based methods*, and *representation-based methods*. In the following, we briefly introduce the representative techniques for each category.

4.2.1. Graph kernel-based methods

A graph kernel is a function measuring the similarity of two graphs. To that end, it represents each graph as a vector representation of graph sub-structures and compares two graphs using an inner product of vector representations. Such graph sub-structures include shortest paths, random walks, and subtree patterns.

Graph kernel-based methods use graph kernels' vector representations of graphs as graph embedding. For instance, Shortest Path (SP) graph kernel (Borgwardt and Kriegel, 2005) uses shortest paths as sub-structures. Suppose graph G is decomposed into n shortest paths and a triplet (s_i, e_i, l_i) represents the labels of the starting and ending nodes (i.e., s_i and e_i) and the length (i.e., l_i of the ith path. G is represented as n-dimensional vector where the ith dimension is the frequency of the ith triplet occurring in G.

Moreover, *Weisfeiler–Lehman (WL)* kernel (Shervashidze et al., 2011) uses subtree patterns, called *Weisfeiler–Lehman* subtrees, as sub-structures. To compute the subtree pattern, we conduct a relabeling iteration process on a labeled graph. At each iteration, a multiset of labels is generated for each node based on the label of the node and the labels of its neighbors. The generated multiset of labels denotes a subtree pattern, which is then used for the next iteration. Suppose n iterations are conducted on graph G. G's embedding contains n blocks where the ith dimension in jth block is the frequency of ith label assigned to a node in the jth iteration. Furthermore, *Random Walks (RW)* kernel (Vishwanathan et al., 2010) measures the similarity of two graphs by counting the number of their matching walks, i.e., random walks are used as sub-structures. A graph is represented as a vector containing the frequency of the matching walks.

4.2.2. Deep learning-based methods

Deep learning-based graph embedding applies deep learning models on graphs. *Graph2vec* is an unsupervised neural embedding framework to learn representations of whole graphs as feature vectors of a fixed length (Narayanan et al., 2017). Considering a graph as a document, it first transforms a graph into the sequence of words. Each word corresponds to the rooted subgraph of degree n of any node of the graph (i.e., the subgraph including all nodes reachable in n hops from the node). Next, it trains doc2vec skip gram model (Mikolov et al., 2013) to learn representations of the word sequence (i.e., graph embeddings).

GL2Vec extends Graph2Vec by incorporating line graphs to deal with edge features (Chen and Koga, 2019). The line graph $L(G)$ of a graph G is a graph mapping every edge in G to a node in $L(G)$ such that the nodes in $L(G)$ hold the edge features in G as the node labels. If G does not have edge labels, a node in $L(G)$ is assigned the degree of the corresponding edge in G as the node label. *GL2Vec* (1) applies *Graph2Vec* to G to derive the embedding of G, (2) applies *Graph2Vec* to $L(G)$ to create the embedding of $L(G)$, and (3) append the embedding of G and the embedding of $L(G)$ to derive the final embedding.

4.2.3. Representation-based methods

Representation-based graph embeddings use the statistical properties of a graph to generate a graph signature vector to represent the graph. Cai and Wang (2018) propose a graph representation based on local information for non-attributed graphs, called *Local Degree Profile (LDP)*. For a graph $G = (V, E)$, the approach extracts features for each node in the following way. For each $v \in V$, $DN(v)$ denotes the multiset of the degrees of all neighboring nodes of v, i.e., $DN(v) = [degree(u) \mid (u, v) \in E]$. Each node feature $F(v)$ contains the node's degree information and its 1-neighborhood, i.e., $F(v)$=$(degree(v), min(DN(v)), max(DN(v)), mean(DN(v)), std(DN(v)))$. The node features are aggregated by applying either a histogram or an empirical distribution operation to produce *LDP*.

An approach proposed in de Lara and Pineau (2018) uses the information from the Laplacian matrix and eigenvalues of a graph to generate its embeddings, called *Spectral Features (SF)*. Let $G = (V, E)$ be an undirected and unweighted graph, $A \in \{0, 1\}^{|V| \times |V|}$ its adjacency matrix w.r.t. an arbitrary indexing of the nodes, and $D = diag(A)$ the matrix of node degrees. The normalized Laplacian of G is defined as $\mathcal{L} = I - D^{-1/2} A D^{-1/2}$. SF is the k smallest positive eigenvalues of $\mathcal{L}$ in ascending order. If the graph has less than k nodes, right zero padding is used to produce a vector of appropriate dimensions.

Verma and Zhang (2017) suggest a more advanced approach to transform a graph $G = (V, E)$ with a high-dimensional sparse representation into a histogram on the dense biharmonic graph kernel. The authors define *Family of Graph Spectral Distance (FGSD)* for each node pair, i.e., $S(x, y)$ for any $(x, y) \in V \times V$, and compute a graph spectrum $\mathcal{R}$ as a multiset of the FGSDs for all node pairs, i.e., $\mathcal{R} = [S(x, y) \mid \forall (x, y) \in V]$. An embedding $\mathcal{F}$ of G is computed as the histogram of $\mathcal{R}$.

However, FGSD does not capture graph features at different scales of graph sizes and has quadratic time complexity, hampering its application to large graphs. Tsitsulin et al. (2018) propose the *Network Laplacian Spectral Descriptor (NetLSD)* that transforms graphs to compact graph signatures based on the heat or wave kernel of the Laplacian, which inherit the formal properties of the Laplacian spectrum.

Some of the existing approaches use characteristic functions defined on graph nodes to describe the distribution of node features. For instance, Wang et al. (2021) use a diffusion-wavelet-based method as a characteristic function to capture topological similarity. The authors propose a graph embedding (called *WaveletCharacteristic*) based on the aggregation of Euclidean node embeddings calculated by the characteristic functions.

In the experiments explained in the following section, we use the aforementioned graph embedding techniques to implement graph embedding, including *Graph2Vec* (Narayanan et al., 2017), *GL2Vec* (Chen and Koga, 2019), *Local Degree Profile (LDP)* (Cai and Wang, 2018), *Spectral Features (SF)* (de Lara and Pineau, 2018), *Family of Graph Spectral Distance (FGSD)* (Verma and Zhang, 2017), *Network Laplacian Spectral Descriptor (NetLSD)* (Tsitsulin et al., 2018), and *WaveletCharacteristic* (Wang et al., 2021).

4.3. End-to-end pipeline

An end-to-end depiction of our predictive pipeline is shown in Fig. 8. Based on the process execution, we determine features and target variables. The process executions need to be split into a training and testing set. When performing a training/testing split later, some process executions could end up in both the training and testing set. Subsequently, we extract subgraphs of (a) given size(s). The corresponding target variable of a subgraph is always the target variable of the (timewise) last event of this subgraph. In general, we allow for different ways of extracting subgraphs, i.e., one can choose only the subgraphs that include the process execution start or exclude the last subgraph. The subgraphs are either directly used as an input for a graph neural network or first embedded into a vector which is used as an input for a tabular machine learning technique. The models are trained on the training set and evaluated on the testing set. Subsequently, these trained models can be used in a streaming setting with ongoing process executions.

5. Evaluation

In this section, we evaluate our proposed methods of preserving object-centric structures through direct graph representations or graph embeddings for machine learning tasks in process mining. First, we apply our methods to a synthetic order-management event log as a proof of concept and show the full end-to-end pipeline from an object-centric event log to predictions. Second, we use a real-life loan application process to measure the amount of information contained in

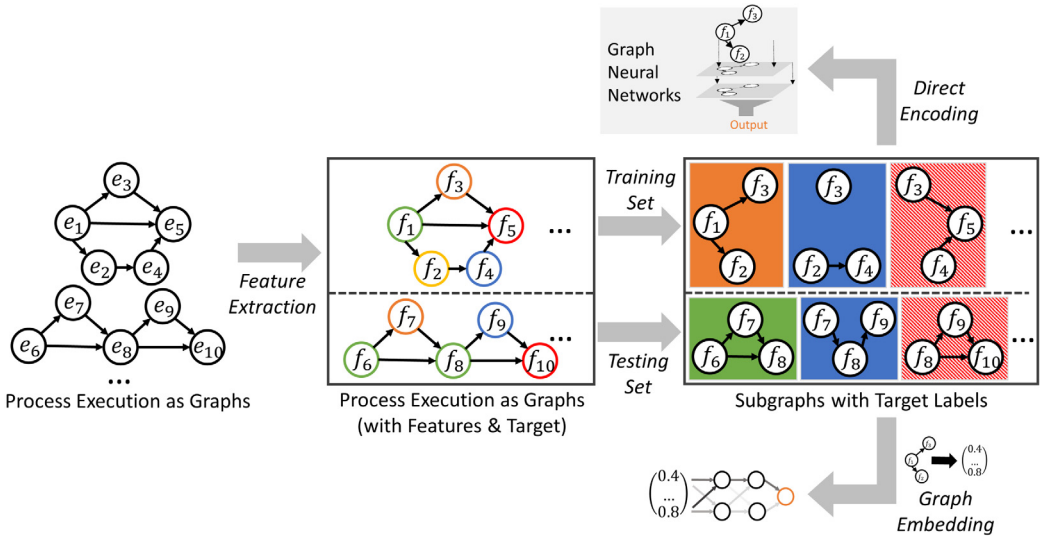

Fig. 8. Example of the end-to-end predictive pipeline: We calculate features including target variables from the process executions. To avoid information leakage, we already split the process executions into training and testing set. Subsequently, we extract subgraphs from the process executions. We either directly train a GNN on the subgraphs or embed them and train, e.g., a multi-layer perceptron.

the graph structure of object-centric event data across three different prediction tasks. We do this by applying graph neural networks and the previously introduced embedding techniques to graph-based and flattened process executions. Furthermore, we assess the best-suited graph embedding techniques to preserve this information. Third, we compare the effectiveness of our employed graph embeddings to current state-of-the-art techniques of preserving object-centric structures through manually designed features (Galanti et al., 2023; Adams et al., 2022a).

5.1. General experimental settings

In this section, we introduce the experimental setting that stays consistent across our experiments. We discuss the different steps of Fig. 8, i.e., the feature extraction, training/testing split, subgraph generation, employed graph neural network model, and the parameter choices for the graph embedding techniques. Our experiments are implemented on the basis of OCPA (Adams et al., 2022b)[1] and are publicly available on GitHub.[2]

Feature extraction and subgraph generation. We extract the process executions according to the connected-components extraction introduced in Section 4. We consider different target variables: next activity, next timestamp, and remaining time. To investigate the usefulness of the graph structure, we do not add any features, i.e., the only available information is the graph structure. In that way, we can assess the pure information contained in the graph structure without any features substituting or enriching it. When flattening the event log, the graph structures are squashed into sequences. We split the process execution into training and testing set using a 0.7/0.3 split. We set aside 10% of the training set as validation set. We generate subgraphs with the number of nodes $k = 2$ through $k = 8$. For the classification task of next activity prediction we use accuracy as an evaluation metric, for the regressive tasks of next timestamp and remaining time we use *Mean Absolute Error* (MAE) as an evaluation metric.

Graph neural network. We use a graph neural network to perform a prediction task directly on the graph structure of the process executions. We implement the graph neural network in the Python library *DGL*.[3] We use two graph convolution layers (Kipf and Welling, 2017) and connect the second convolution layer to a dense layer transforming the output of the graph convolution layer to the network output. The number of nodes for the graph neural network is fixed in our setting, i.e., only graphs with k nodes can be used as input. We, therefore, extract a subgraph of the preceding k events for each event, linking this subgraph to the corresponding target variable at this event for multiple values of k.

The graph neural network is also used to make predictions for the structure of the flattened event log. The structure of the flattened event log can be derived by flattening a process execution, i.e., sequentializing its nodes according to the timestamps. The sequence, which can be represented as a graph, is used as an input to the graph neural network. The performance achieved by the graph neural network on the flattened structures is the baseline performance of flattened event logs.

Graph embedding. The Karate Club (Rozemberczki et al., 2020) software package is used to create graph embeddings based on deep learning-based methods (Narayanan et al., 2017; Chen and Koga, 2019) and representation-based methods (Cai and Wang, 2018; de Lara and Pineau, 2018; Verma and Zhang, 2017; Tsitsulin et al., 2018; Wang et al., 2021). We use the default hyperparameter settings of the 1.3 release, while limiting the output vector size to the tenfold of the number of graph nodes when applicable to avoid overfitting. This is not possible for NetLSD and WaveletCharacteristics which constraint the vector size to 1000 and 160 elements. The embeddings are used for predictive tasks by feeding them into either a linear regression or a simple MLP with two hidden layers of five nodes each. We use an output layer of one node for regressive tasks and an output layer with one-hot encoding for classification tasks.

5.2. Synthetic data

We showcase our proposed framework using a synthetic order management event log.[4] This event log consists of 22367 events and 11848 objects. Together, these events and objects span 83 process executions that vary in size from 9 to over 3000 events. We depict an example variant using the visualization proposed in Adams et al. (2022c) obtained from OCπ (Adams and van der Aalst, 2022) in Fig. 9. We show the

[1] https://github.com/ocpm/ocpa
[2] https://github.com/niklasadams/PreservingOCStructures
[3] https://www.dgl.ai/
[4] Provided by http://ocel-standard.org

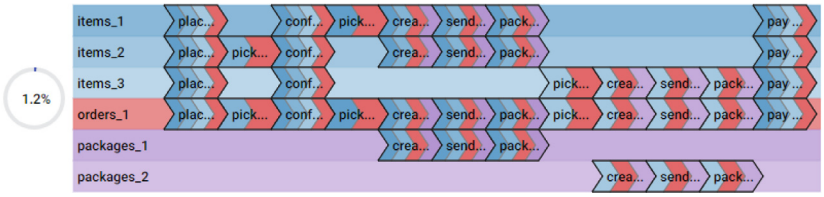

Fig. 9. Variant visualization of one single process execution of the order management event log. Process executions of this log can contain a large number of objects and form large graphs.

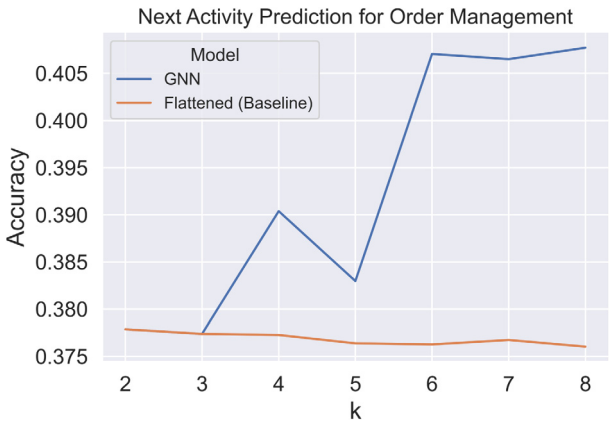

Fig. 10. Accuracy results for the next activity prediction on the order management event log. The graph structure of process executions can be leveraged by the graph neural network.

results for the next activity prediction using the Graph Neural Network on the process executions and the flattened event data in Fig. 10. We observe better or equal accuracy results for any size of the subgraphs k. While the gap between accuracy results is, generally, lower for smaller k values, it increases with growing k. The results show that the graph neural network can infer valuable information for the next activity prediction from the graph structure of the process execution and, therefore, provides the results as they were expected.

5.3. Quantification of the informational value of the graph structures

In this section, we assess the informational value of object-centric graph structures using a real-life event log. By doing so, we are able to quantify the degree of predictive value contained in these structures and single out the most effective technique to preserve these structures. To do so, we compare the predictive performances of three different models:

(1) Graph neural networks using the process execution graphs directly as input.
(2) Regression and Multi-Layer Perceptron (MLP) models using graph embeddings as input.
(3) The baseline performance of flattened event logs with graph neural networks using the flattened process executions as input.

First, we introduce the real-life event data used in this evaluation. Second, we compare the predictive performance on the graph-based process executions with the performance on flattened process executions. At last, we compare the effectiveness of the different techniques.

5.3.1. Experimental setting

We use a real-life event log of a loan application business process containing 507553 events of 67498 objects, spanning 31509 process executions (van Dongen, 2017). This event log describes an application for a loan that is matched with an offer from a financial institution. Each event can be associated with objects of type application and offer. An application can be associated with multiple offers. With only two object types, shows a comparably small degree of object-centricity. Exemplary variant visualizations of process executions are depicted in Fig. 11.

5.3.2. Experimental results

Fig. 12 depicts the performance of the graph neural network, all graph embeddings with regression and MLP, and the baseline of a flattened event log for the three different prediction tasks. The y-axes show the performance while the x-axes show the size of graphs denoted as k. The performances of the graph neural network on object-centric structures and flattened structures are depicted with individual lines, and the performance of the graph embeddings combined with either regression or MLP is depicted using the average with the 95% confidence interval to show the range of different graph embedding techniques. We structure the presentation and discussion of results in two parts: First, we compare the performances of graph-based techniques against the baseline performance on sequential structures of flattened event logs. Second, we compare the performance of graph neural networks and graph embeddings against each other to single out the most effective technique.

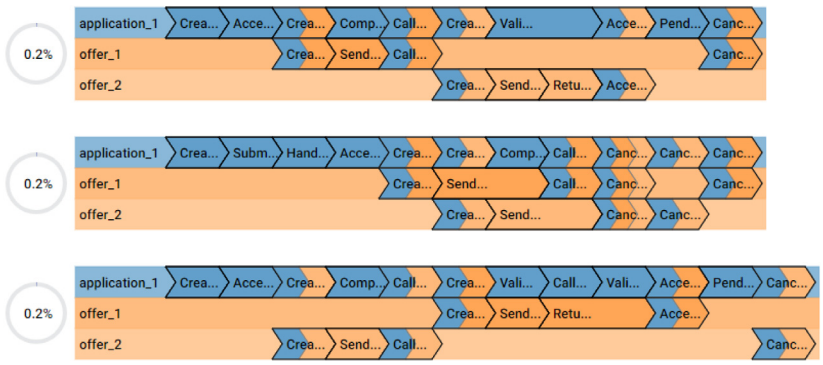

Fig. 11. Variant visualization of one single process execution of the loan application event log. Process executions of this log, generally, contain one application and one or more offers. The corresponding process execution graphs are smaller than the ones of the order management event log.

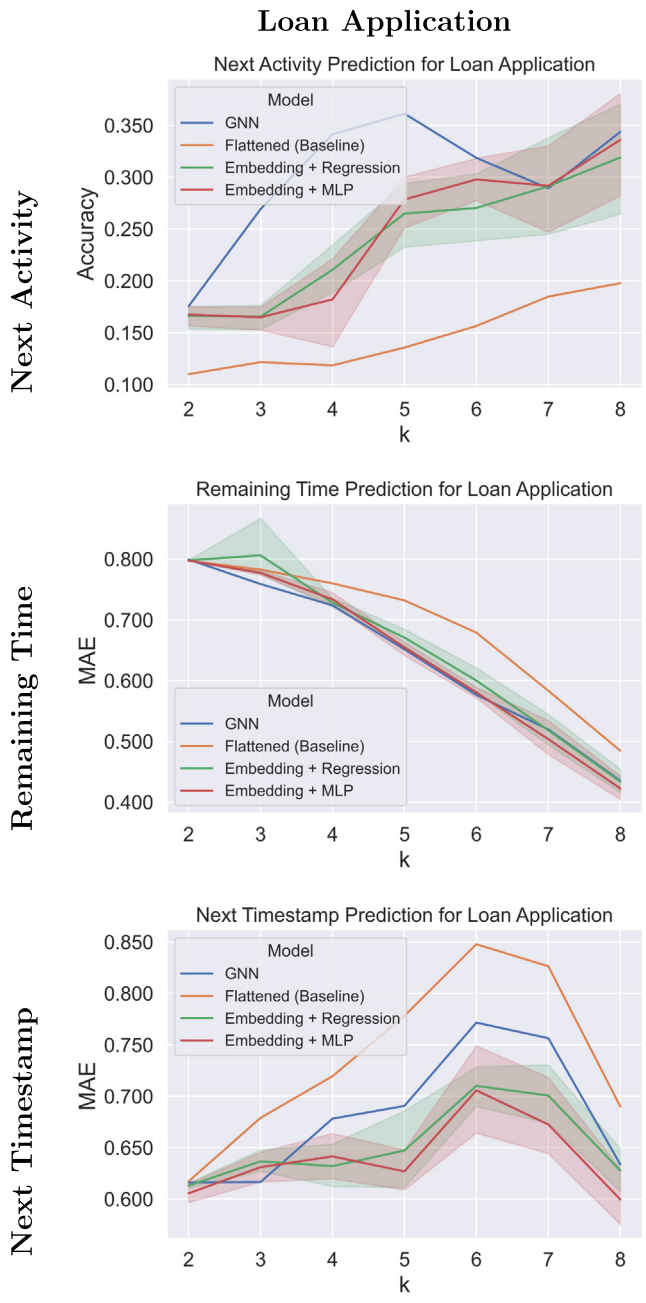

Fig. 12. Predictive performance of different techniques for preserving object-centric structures compared to the baseline performance on the sequential structure of the flattened event log.

Graphs and flat event logs. We used exactly the same graph neural network and altered the input between process execution graphs and flattened process executions to produce the lines labeled with *GNN* and *Flattened (Baseline)* in Fig. 12. The relative difference between the predictive performances, therefore, provides information about the predictive value of the graph structures of the object-centric event log. *For any task, event log, and graph size, the graph structure of the object-centric event log provides additional predictive value over the flattened event log.* Especially for next activity and next timestamp prediction, the graph structure itself carries a large amount of useful information for prediction. For many tasks, the gap between flat and object-centric structures increases with the size of the graph. However, the difference between prediction tasks hints at task-dependent levels of the utility of the structure itself: While the graph structure can be extremely useful for the next activity and next timestamp prediction, e.g., by preserving the current activity, the graph structure itself yields comparably less predictive value for the remaining time prediction. However, enriched with corresponding features of, e.g. current timestamp, the graph structure should be able to achieve significantly better results. The performance of graph embeddings with both regression and MLP confirms this trend. The performance is almost always comparable to graph neural networks, sometimes even better, and the graph embeddings are able to preserve a significant amount of information, even when paired with models as simple as a regression.

Comparing different graph embeddings and graph neural networks. In the previous section, we depicted how using both graph neural networks and graph embeddings for process executions enables the improvement of prediction results by capturing relevant structural information. We, now, compare all eight techniques, i.e., the seven graph embedding techniques and graph neural networks, to single out the most effective technique for preserving graph structures of process executions. The individual results of the graph embedding techniques are depicted in Fig. 13.

Fig. 13 depicts the performance of the seven selected graph embedding techniques for all three prediction tasks. We paired each graph embedding with a regression and an MLP to learn predictions from the graph embedding. The color of the line indicates the technique, and the type of the line indicates whether a regression or an MLP was used to perform the prediction.

Several observations can be made across multiple prediction tasks. First of all, the best results are usually achieved using an MLP as prediction technique. The performance of regression is, however, often quite comparable. The employed graph embedding technique has, unsurprisingly, a greater effect on the results. Across the different prediction tasks, observed patterns in relative performance between embedding techniques stay quite consistent: GL2Vec (Chen and Koga, 2019) and FGSD (Verma and Zhang, 2017) are the best-performing graph embeddings in almost every evaluation. In contrast, NetLSD (Tsitsulin et al., 2018) and SF (de Lara and Pineau, 2018) are among the worst-performing approaches for every evaluation. WaveletCharacteristics (Wang et al., 2021) often show promising results but are also relatively unstable in their performance. LDP (Cai and Wang, 2018) and Graph2Vec (Narayanan et al., 2017) consistently exhibit performances ranking them in the midfield. When employing graph embeddings for object-centric event data, either FGSD or GL2Vec can be the best suited options preserving the most information. The prediction tasks do not have a significant influence of the suitability of an individual graph embedding technique.

To quantify these observations, we aggregate the predictive performance in comparison to the flattened baseline (the relative improvement) of all prediction tasks for all graph embeddings and the graph neural network. We depict the results in Fig. 14. The best-performing techniques are the Graph Neural Network, Wavelet Characteristics, LDP, GL2Vec, and FGSD. Due to the consistency of FGSD's performance as well as the good performance for larger graphs, its average performance is the best. We average the relative improvement of the prediction when using FGSD and retrieve a value of 40.8%. With this performance, FGSD even almost defeats the graph neural network which achieves an average value of 41.7% This also provides an answer to our first research question **RQ1**: *Through flattening, structural information that is equivalent to an improvement of 41.7% across the three predictive targets is lost.* The results of these experiments also already point out that FGSD is the most effective among our investigated techniques. When looking at the inherent algorithmic properties of FGSD, this superior performance might be due to the locality of graph features that are invariant under isomorphism (Verma and Zhang, 2017): As the authors elaborate, FGSD graph embeddings capture the sub-structure similarity between two graphs. This allows learning association for certain patterns (i.e. subgraphs) that appear in the process executions.

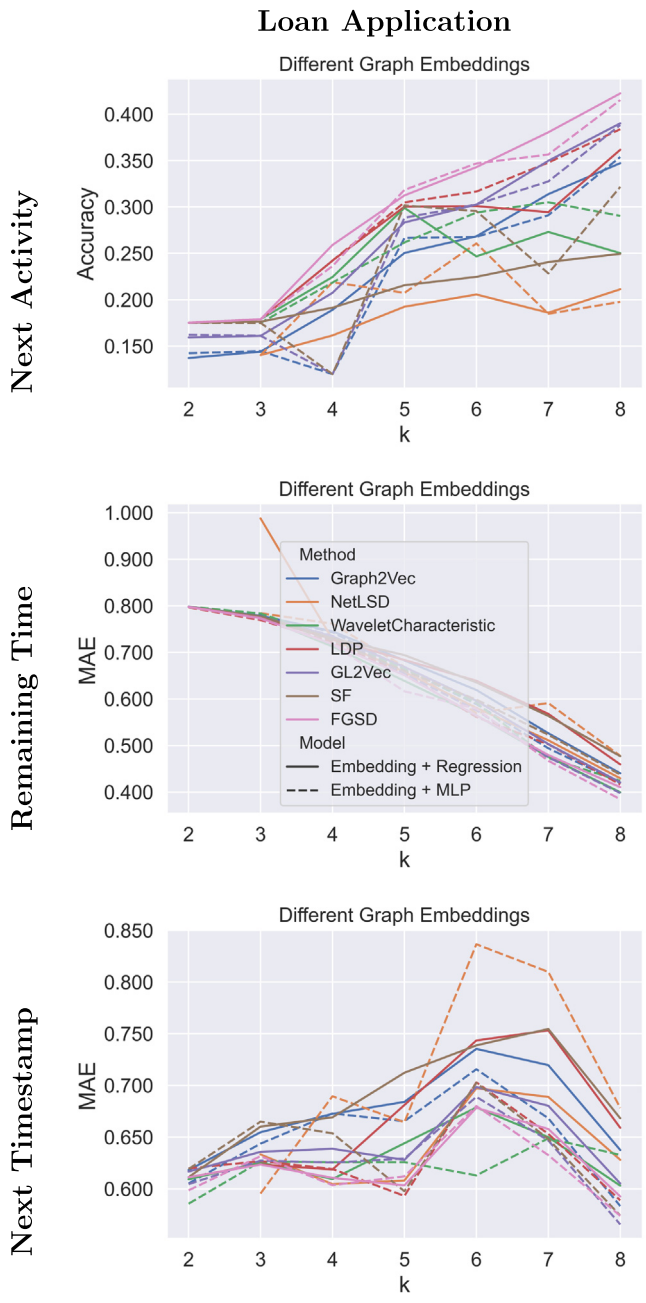

Fig. 13. Performance of different graph embedding techniques for the three different prediction tasks and two different event logs. The graphs share the legends of the middle plot.

Using these associations, the appearance of a pattern in one process execution can be linked to the target variable. Without such a sub-structure similarity, the graph embedding would only be able to distinguish between different graphs, effectively only learning loss minimizations for individual graphs without transferring common characteristics. Our experiments validate the statement of the authors that FGSD can effectively learn sub-structure similarity. This is an interesting pointer to future employment and the development of tailored graph embeddings for process executions. Due to the possibility of adding node labels to the graph and applying an FGSD embedding (Verma and Zhang, 2017), FGSD can be a promising candidate for end-to-end predictive process monitoring pipelines.

In the next part of our experiment, we analyze whether the graph embeddings are able to capture information that current approaches encoding object information through manually designed features cannot capture.

5.4. Graph embedding benchmark

In this section, we compare the effectiveness of our employed graph embeddings to state-of-the-art approaches of encoding object information for machine learning tasks. As discussed in Section 2, current approaches use features to capture object information. Galanti et al. (2023) propose to use the number of objects of one type and the relative share of objects of one type that have run through each activity as event-level features to encode the object information. Adams et al. (2022a) also propose to use the number of objects of a type as a feature capturing the object perspective.

Experimental setting. Our experimental setting aims to quantify how much added information is contained in the graph embeddings that is not captured by current feature-based methods (Adams et al., 2022a; Galanti et al., 2023). To this end, we extract these features for the process executions, flatten them, and obtain a two-dimensional array with all feature values for each event. We proceed in two different ways: First, we append a vector of random noise to the features. Second, instead of the random noise vector, we append a graph embedding vector to the features. We train a predictive model on both obtained feature sets and compare their predictive performance of them. The relative improvement of the prediction with respect to the naive baseline is the added value of graph embedding compared to state-or-the-art feature-based embeddings. We depict the full experimental framework in Fig. 15. We perform this comparison for a total of 147 MLPs that are composed as follows: For each of the three targets (next activity, remaining time, and next timestamp) and each subgraph size $k = 2$ through $k = 8$ we train and evaluate seven different models. These seven different models are composed of one linear/logistic regression and six MLP Regressors/Classifiers with either one layer of 20, 18, or 16 nodes or two layers with (20,10), (18,9), or (16,8) nodes. For the graph embeddings, we use FGSD as it have shown the best performance in previous experiments.

Experimental results. The results of our experiments are depicted in Fig. 16. Across all subgraph sizes, we observe an improved performance when using the graph embeddings. The predictive improvement grows with larger graphs. When averaging over all subgraph sizes, predictive targets, and models we achieve an average improvement of 2.43% whereas the average improvement for the largest subgraph size, $k = 8$, lies at 4.42%. This is the amount of predictive information that is not captured by the features aiming to encode object information. When considering the relatively low amount of object-centricity in the loan application event log, i.e., only two object types and only one type with more than one object per execution, these results show a promising basis for the improvement of machine learning tasks in process mining by incorporating object-centric event logs and graph embeddings. These experiments provide an answer to our research question **RQ2**: The best technique for capturing the graph structures of object-centric event data is an FGSD Graph Embedding with an average improvement of 2.43% over the current state-of-the-art for all subgraph sizes and up to 4.42% for the largest evaluated subgraph size.

5.5. Threats to validity

In this section, we critically discuss our evaluation and the derived answers to our research questions. We split this into two parts: First, we discuss the validity of our experimental setup and, second, we assess the generalizability of our results.

Through our experimental setup, we address several typical issues that arise with evaluations in machine learning: Overfitting, underfitting, and data preparation issues. We avoid overfitting by employing a

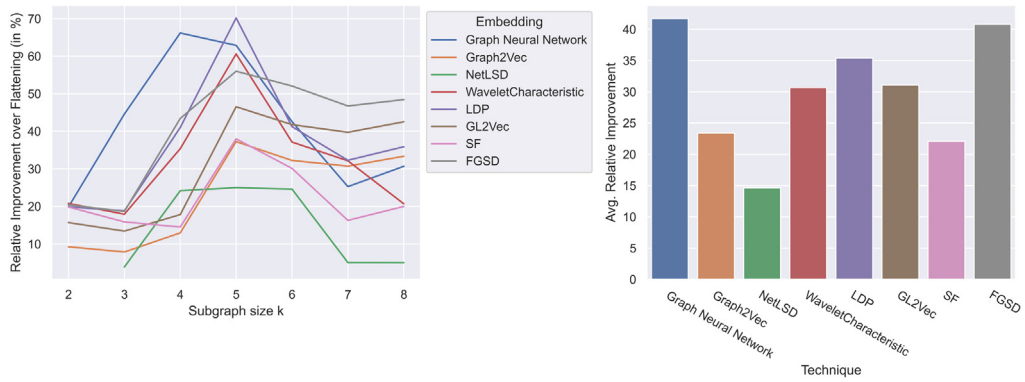

Fig. 14. Performance relative to the flattened baseline for all embedding techniques and the GNN across all targets and subgraph sizes.

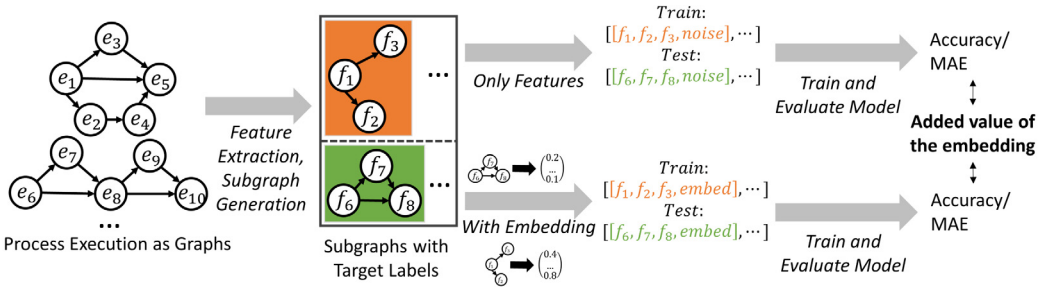

Fig. 15. Our experimental framework to assess the added benefit of graph embeddings compared to any features that aim to capture object information. We evaluate against state-of-the-art features.

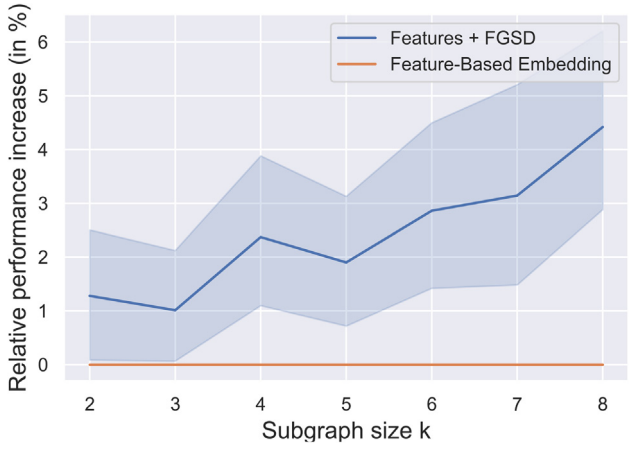

Fig. 16. Relative performance increase of different predictive models across three target variables when adding an FGSD graph embedding to state-of-the-art features capturing object information.

train-test split as well as using a validation split on the training data. Underfitting can be ruled out due to the comparison to the baseline performance of flattening. In the data preparation phase, we normalize the data based on the training set. Furthermore, we ensure that all prediction instances of the same process execution are assigned to either the train or test set to prevent information leakage. It is also notable that we eliminate confounding factors by isolating the effect of only one component in our experimental setup: The only difference between the data sets in the first part of the evaluation is the application of flattening, in the second part of the evaluation it is replacing random noise by the graph embedding vector. Therefore, the change in performance can clearly be associated with these components.

When looking at the generalizability of our results, we investigate two different perspectives: The generalizability of the experiments for the specific data sets and the generalizability of our conclusions for the research questions. On the specific data sets, we ran a large number of experiments: seven subgraph sizes, three target variables, eight graph embeddings, and two prediction models for the first part of the evaluation, and seven subgraph sizes, three target variables, and seven different models for the second part of the evaluation. The results generated for all these experiments support our answers to the research questions. The only limiting threat to validity is the number of data sets: The results were generated using two data sets. Once more data sets become publicly available, the validity of the results should be confirmed.

6. Conclusion

Existing techniques for machine learning tasks in process mining flatten object-centric event data into traditional event data. The flattening process eliminates important structural information about the interaction between objects and subprocesses from the event data. In this paper, we introduced a general approach to preserve this structural information for machine learning tasks by using direct graph encodings or graph embeddings. We show how much information is captured in the structure – which is eliminated when flattening – by comparing the results of three different prediction tasks on the graph structures of object-centric event data and the sequential structures of flattened real-life event data. Through our experiments, we answer both research questions: **RQ1:** Flattening eliminates structural information that is equivalent to an average 41.7% performance increase across all predictive tasks. **RQ2:** These results are achieved when using graph neural networks, closely followed by the performance of the best-performing graph embedding FGSD. To compare against existing approaches, we assessed how much structural information can be captured by FGSD that cannot be captured by current approaches of encoding object information through manually-designed features. The graph embedding of FGSD contains structural information that is equivalent to an average 2.43% performance increase across predictive tasks.

6.1. Future work

Our work provides a foundation for three future research directions: First, based on our findings, current predictive analytics frameworks can be adapted to object-centric event data such that structure is considered and predictions are improved. This is already relevant when extracting the data in an object-centric way. Second, the best-performing graph embedding techniques can be adapted according to assumptions of object-centric process mining to develop even better-specialized techniques. Furthermore, for these techniques, computational demands and resource efficiency specifically in the setting of object-centric process mining can be evaluated between models. Third, the demonstrated shortcoming of current approaches to encode object-centricity into predictive process monitoring frameworks leads to research questions on optimal feature sets to encode object-centricity. Even though current features can contain a lot of information about individual object flow, they were missing information that graph embeddings could capture. A systematic evaluation could assess which features and which combination of features could work best to close this gap, effectively functioning as a graph embedding.

6.2. Implications

Our paper has exposed a critical weak point of current machine learning frameworks for process mining: Important structural information is lost by flattening the event log. This has far-reaching implications for research and practical deployments: First, state-of-the-art machine learning frameworks, e.g., predictive process monitoring or clustering frameworks, should better caption object-centricity, either through graph embeddings, graph neural networks, or through more dedicated features capturing object-centricity. Second, practical deployments should adapt their event log extraction to accurately capture object-centric event logs and, therefore, object interactions. These adoptions would increase the performance of operational support with machine learning models, increasing the value provided to customers.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Code and data are available in the public Github repository and the Python library.

Acknowledgments

We thank the Alexander von Humboldt (AvH) Stiftung, Germany for supporting our research.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.engappai.2023.106764.

References

van der Aalst, W.M.P., 2016. Process Mining - Data Science in Action, second ed. Springer, http://dx.doi.org/10.1007/978-3-662-49851-4.

van der Aalst, W.M.P., 2019. Object-centric process mining: Dealing with divergence and convergence in event data. In: Ölveczky, P.C., Salaün, G. (Eds.), Software Engineering and Formal Methods - 17th International Conference, Proceedings. SEFM 201, Oslo, Norway, September 18–20, 2019, In: Lecture Notes in Computer Science, vol. 11724, Springer, pp. 3–25. http://dx.doi.org/10.1007/978-3-030-30446-1_1.

van der Aalst, W.M.P., Artale, A., Montali, M., Tritini, S., 2017. Object-centric behavioral constraints: Integrating data and declarative process modelling. In: Artale, A., Glimm, B., Kontchakov, R. (Eds.), Proceedings of the 30th International Workshop on Description Logics, July 18–21, 2017. In: CEUR Workshop Proceedings, 1879, CEUR-WS.org, Montpellier, France.

van der Aalst, W.M.P., Berti, A., 2020. Discovering object-centric Petri nets. Fundam. Inform. 175 (1–4), 1–40. http://dx.doi.org/10.3233/FI-2020-1946.

van der Aalst, W.M.P., et al., 2011. Process mining manifesto. In: Daniel, F., Barkaoui, K., Dustdar, S. (Eds.), Business Process Management Workshops - International Workshops, Revised Selected Papers, Part I. BPM 2011, Clermont-Ferrand, France, August 29, 2011, In: Lecture Notes in Business Information Processing, vol. 99, Springer, pp. 169–194. http://dx.doi.org/10.1007/978-3-642-28108-2_19.

Adams, J.N., Park, G., Levich, S., Schuster, D., van der Aalst, W.M.P., 2022a. A framework for extracting and encoding features from object-centric event data. In: Troya, J., Medjahed, B., Piattini, M., Yao, L., Fernández, P., Ruiz-Cortés, A. (Eds.), Service-Oriented Computing - 20th International Conference, Proceedings. ICSOC 2022, Seville, Spain, November 29 - December 2, 2022, In: Lecture Notes in Computer Science, vol. 13740, Springer, pp. 36–53. http://dx.doi.org/10.1007/978-3-031-20984-0_3.

Adams, J.N., Park, G., van der Aalst, W.M., 2022b. ocpa: A Python library for object-centric process analysis. Softw. Impact. 14, 100438. http://dx.doi.org/10.1016/j.simpa.2022.100438.

Adams, J.N., Schuster, D., Schmitz, S., Schuh, G., van der Aalst, W.M.P., 2022c. Defining cases and variants for object-centric event data. In: Burattin, A., Polyvyanyy, A., Weber, B. (Eds.), 4th International Conference on Process Mining. ICPM 2022, Bolzano, Italy, October 23–28, 2022, IEEE, pp. 128–135. http://dx.doi.org/10.1109/ICPM57379.2022.9980730.

Adams, J.N., van der Aalst, W.M.P., 2021. Precision and fitness in object-centric process mining. In: Ciccio, C.D., Francescomarino, C.D., Soffer, P. (Eds.), 3rd International Conference on Process Mining. ICPM 2021, Eindhoven, the Netherlands, October 31 - Nov. 4, 2021, IEEE, pp. 128–135. http://dx.doi.org/10.1109/ICPM53251.2021.9576886.

Adams, J.N., van der Aalst, W.M.P., 2022. OCπ: Object-centric process insights. In: Bernardinello, L., Petrucci, L. (Eds.), Application and Theory of Petri Nets and Concurrency - 43rd International Conference, Proceedings. PETRI NETS 2022, Bergen, Norway, June 19-24, 2022, In: Lecture Notes in Computer Science, vol. 13288, Springer, pp. 139–150. http://dx.doi.org/10.1007/978-3-031-06653-5_8.

Ali, A., Zhu, Y., Zakarya, M., 2022. Exploiting dynamic spatio-temporal graph convolutional neural networks for citywide traffic flows prediction. Neural Netw. 145, 233–247. http://dx.doi.org/10.1016/j.neunet.2021.10.021.

Augusto, A., Conforti, R., Dumas, M., Rosa, M.L., Maggi, F.M., Marrella, A., Mecella, M., Soo, A., 2019. Automated discovery of process models from event logs: Review and benchmark. IEEE Trans. Knowl. Data Eng. 31 (4), 686–705. http://dx.doi.org/10.1109/TKDE.2018.2841877.

Babai, L., 2016. Graph isomorphism in quasipolynomial time [extended abstract]. In: Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing. STOC '16, Association for Computing Machinery, New York, NY, USA, pp. 684–697. http://dx.doi.org/10.1145/2897518.2897542.

Bauer, M., van der Aa, H., Weidlich, M., 2019. Estimating process conformance by trace sampling and result approximation. In: Hildebrandt, T.T., van Dongen, B.F., Röglinger, M., Mendling, J. (Eds.), Business Process Management - 17th International Conference, Proceedings. BPM 2019, Vienna, Austria, September 1–6, 2019, In: Lecture Notes in Computer Science, vol. 11675, Springer, pp. 179–197. http://dx.doi.org/10.1007/978-3-030-26619-6_13.

Berti, A., van der Aalst, W.M.P., 2019. Extracting multiple viewpoint models from relational databases. In: Ceravolo, P., van Keulen, M., López, M.T.G. (Eds.), Data-Driven Process Discovery and Analysis - 8th IFIP WG 2.6 International Symposium, SIMPDA 2018, Seville, Spain, December 13–14, 2018, and 9th International Symposium, Revised Selected Papers. SIMPDA 2019, Bled, Slovenia, September 8, 2019, In: Lecture Notes in Business Information Processing, vol. 379, Springer, pp. 24–51. http://dx.doi.org/10.1007/978-3-030-46633-6_2.

Borgwardt, K.M., Kriegel, H., 2005. Shortest-path kernels on graphs. In: Proceedings of the 5th IEEE International Conference on Data Mining. ICDM 2005, Houston, Texas, USA, 27-30 November 2005, IEEE Computer Society, pp. 74–81. http://dx.doi.org/10.1109/ICDM.2005.132.

Cai, C., Wang, Y., 2018. A simple yet effective baseline for non-attributed graph classification. http://dx.doi.org/10.48550/ARXIV.1811.03508, arXiv.

Cai, H., Zheng, V.W., Chang, K.C., 2018. A comprehensive survey of graph embedding: Problems, techniques, and applications. IEEE Trans. Knowl. Data Eng. 30 (9), 1616–1637. http://dx.doi.org/10.1109/TKDE.2018.2807452.

Calvanese, D., Montali, M., Estañol, M., Teniente, E., 2014. Verifiable UML artifact-centric business process models. In: Li, J., Wang, X.S., Garofalakis, M.N., Soboroff, I., Suel, T., Wang, M. (Eds.), Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management. CIKM 2014, Shanghai, China, November 3–7, 2014, ACM, pp. 1289–1298. http://dx.doi.org/10.1145/2661829.2662050.

Calvanese, D., Montali, M., Syamsiyah, A., van der Aalst, W.M.P., 2015. Ontology-driven extraction of event logs from relational databases. In: Reichert, M., Reijers, H.A. (Eds.), Business Process Management Workshops - 13th International Workshops, Revised Papers. BPM 2015, Innsbruck, Austria, August 31 - September 3, 2015, In: Lecture Notes in Business Information Processing, vol. 256, Springer, pp. 140–153. http://dx.doi.org/10.1007/978-3-319-42887-1_12.

Carmona, J., van Dongen, B.F., Solti, A., Weidlich, M., 2018. Conformance Checking - Relating Processes and Models. Springer, http://dx.doi.org/10.1007/978-3-319-99414-7.

Castellanos, M., Salazar, N., Casati, F., Dayal, U., Shan, M., 2006. Predictive business operations management. Int. J. Comput. Sci. Eng. 2 (5/6), 292–301. http://dx.doi.org/10.1504/IJCSE.2006.014772.

Chamorro, A.E.M., Nepomuceno-Chamorro, I.A., Resinas, M., Ruiz-Cortés, A., 2022. Updating prediction models for predictive process monitoring. In: Franch, X., Poels, G., Gailly, F., Snoeck, M. (Eds.), Advanced Information Systems Engineering - 34th International Conference, Proceedings. CAiSE 2022, Leuven, Belgium, June 6–10, 2022, In: Lecture Notes in Computer Science, vol. 13295, Springer, pp. 304–318. http://dx.doi.org/10.1007/978-3-031-07472-1_18.

Chen, H., Koga, H., 2019. GL2vec: Graph embedding enriched by line graphs with edge features. In: Gedeon, T., Wong, K.W., Lee, M. (Eds.), Neural Information Processing - 26th International Conference, Proceedings, Part III. ICONIP 2019, Sydney, NSW, Australia, December 12–15, 2019, In: Lecture Notes in Computer Science, vol. 11955, Springer, pp. 3–14. http://dx.doi.org/10.1007/978-3-030-36718-3_1.

Chiorrini, A., Diamantini, C., Mircoli, A., Potena, D., 2021. Exploiting instance graphs and graph neural networks for next activity prediction. In: Munoz-Gama, J., Lu, X. (Eds.), Process Mining Workshops - International Workshops, Revised Selected Papers. ICPM 2021, Eindhoven, the Netherlands, October 31 - November 4, 2021, In: Lecture Notes in Business Information Processing, vol. 433, Springer, pp. 115–126. http://dx.doi.org/10.1007/978-3-030-98581-3_9.

Cohn, D., Hull, R., 2009. Business artifacts: A data-centric approach to modeling business operations and processes. IEEE Data Eng. Bull. 32 (3), 3–9.

Ding, C., Wen, S., Ding, W., Liu, K., Belyaev, E., 2022. Temporal segment graph convolutional networks for skeleton-based action recognition. Eng. Appl. Artif. Intell. 110, 104675. http://dx.doi.org/10.1016/j.engappai.2022.104675.

van Dongen, B., 2017. BPI Challenge 2017. http://dx.doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b.

Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.A., 2018. Fundamentals of Business Process Management, second ed. Springer, http://dx.doi.org/10.1007/978-3-662-56509-4.

Eldin, A.N., Assy, N., Kobeissi, M., Baudot, J., Gaaloul, W., 2022. Enabling multi-process discovery on graph databases. In: Sellami, M., Ceravolo, P., Reijers, H.A., Gaaloul, W., Panetto, H. (Eds.), Cooperative Information Systems - 28th International Conference, Proceedings. CoopIS 2022, Bozen-Bolzano, Italy, October 4–7, 2022, In: Lecture Notes in Computer Science, vol. 13591, Springer, pp. 112–130. http://dx.doi.org/10.1007/978-3-031-17834-4_7.

ElMaraghy, H., Azab, A., Schuh, G., Pulz, C., 2009. Managing variations in products, processes and manufacturing systems. CIRP Ann. 58 (1), 441–446. http://dx.doi.org/10.1016/j.cirp.2009.04.001.

Esser, S., Fahland, D., 2021. Multi-dimensional event data in graph databases. J. Data Semant. 10 (1–2), 109–141. http://dx.doi.org/10.1007/s13740-021-00122-1.

Evermann, J., Rehse, J., Fettke, P., 2017. Predicting process behaviour using deep learning. Decis. Support Syst. 100, 129–140. http://dx.doi.org/10.1016/j.dss.2017.04.003.

Fahland, D., 2019. Describing behavior of processes with many-to-many interactions. In: Donatelli, S., Haar, S. (Eds.), Application and Theory of Petri Nets and Concurrency - 40th International Conference, Proceedings. PETRI NETS 2019, Aachen, Germany, June 23–28, 2019, In: Lecture Notes in Computer Science, vol. 11522, Springer, pp. 3–24. http://dx.doi.org/10.1007/978-3-030-21571-2_1.

Fahland, D., 2022. Process mining over multiple behavioral dimensions with event knowledge graphs. In: van der Aalst, W.M.P., Carmona, J. (Eds.), Process Mining Handbook. In: Lecture Notes in Business Information Processing, vol. 448, Springer, pp. 274–319. http://dx.doi.org/10.1007/978-3-031-08848-3_9.

Fahland, D., de Leoni, M., van Dongen, B.F., van der Aalst, W.M.P., 2011. Conformance checking of interacting processes with overlapping instances. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (Eds.), Business Process Management - 9th International Conference, Proceedings. BPM 2011, Clermont-Ferrand, France, August 30 - September 2, 2011, In: Lecture Notes in Computer Science, vol. 6896, Springer, pp. 345–361. http://dx.doi.org/10.1007/978-3-642-23059-2_26.

Francescomarino, C.D., Ghidini, C., 2022. Predictive process monitoring. In: van der Aalst, W.M.P., Carmona, J. (Eds.), Process Mining Handbook. In: Lecture Notes in Business Information Processing, vol. 448, Springer, pp. 320–346. http://dx.doi.org/10.1007/978-3-031-08848-3_10.

Galanti, R., de Leoni, M., Navarin, N., Marazzi, A., 2023. Object-centric process predictive analytics. Expert Syst. Appl. 213, 119173. http://dx.doi.org/10.1016/j.eswa.2022.119173.

Georgakopoulos, D., Hornick, M.F., Sheth, A.P., 1995. An overview of workflow management: From process modeling to workflow automation infrastructure. Distrib. Parallel Databases 3 (2), 119–153. http://dx.doi.org/10.1007/BF01277643.

Ghahfarokhi, A.F., Park, G., Berti, A., van der Aalst, W.M.P., 2021. OCEL: a standard for object-centric event logs. In: Bellatreche, L., Dumas, M., Karras, P., Matulevicius, R., Awad, A., Weidlich, M., Ivanovic, M., Hartig, O. (Eds.), New Trends in Database and Information Systems - Short Papers, Doctoral Consortium and Workshops: DOING, SIMPDA, MADEISD, MegaData, CAoNS, Proceedings. ADBIS 2021, Tartu, Estonia, August 24–26, 2021, In: Communications in Computer and Information Science, vol. 1450, Springer, pp. 169–175. http://dx.doi.org/10.1007/978-3-030-85082-1_16.

Ghilardi, S., Gianola, A., Montali, M., Rivkin, A., 2020. Petri nets with parameterised data - modelling and verification. In: Fahland, D., Ghidini, C., Becker, J., Dumas, M. (Eds.), Business Process Management - 18th International Conference, Proceedings. BPM 2020, Seville, Spain, September 13–18, 2020, In: Lecture Notes in Computer Science, vol. 12168, Springer, pp. 55–74. http://dx.doi.org/10.1007/978-3-030-58666-9_4.

Ingvaldsen, J.E., Gulla, J.A., 2007. Preprocessing support for large scale process mining of SAP transactions. In: ter Hofstede, A.H.M., Benatallah, B., Paik, H. (Eds.), Business Process Management Workshops, International Workshops, BPI, BPD, CBP, ProHealth, RefMod, Semantics4ws, Revised Selected Papers. BPM 2007, Brisbane, Australia, September 24, 2007, In: Lecture Notes in Computer Science, vol. 4928, Springer, pp. 30–41. http://dx.doi.org/10.1007/978-3-540-78238-4_5.

Jalali, A., 2022. Object type clustering using Markov directly-follow multigraph in object-centric process mining. IEEE Access 1. http://dx.doi.org/10.1109/ACCESS.2022.3226573.

Jans, M., Soffer, P., 2017. From relational database to event log: Decisions with quality impact. In: Teniente, E., Weidlich, M. (Eds.), Business Process Management Workshops - International Workshops, Revised Papers. BPM 2017, Barcelona, Spain, September 10–11, 2017, In: Lecture Notes in Business Information Processing, vol. 308, Springer, pp. 588–599. http://dx.doi.org/10.1007/978-3-319-74030-0_46.

Kipf, T.N., Welling, M., 2017. Semi-supervised classification with graph convolutional networks. In: 5th International Conference on Learning Representations, Conference Track Proceedings. ICLR 2017, Toulon, France, April 24–26, 2017, OpenReview.net, URL https://openreview.net/forum?id=SJU4ayYgl.

Kondor, R., Borgwardt, K.M., 2008. The skew spectrum of graphs. In: Cohen, W.W., McCallum, A., Roweis, S.T. (Eds.), Machine Learning, Proceedings of the Twenty-Fifth International Conference. ICML 2008, Helsinki, Finland, June 5–9, 2008, In: ACM International Conference Proceeding Series, 307, ACM, pp. 496–503. http://dx.doi.org/10.1145/1390156.1390219.

Kondor, R., Shervashidze, N., Borgwardt, K.M., 2009. The graphlet spectrum. In: Danyluk, A.P., Bottou, L., Littman, M.L. (Eds.), Proceedings of the 26th Annual International Conference on Machine Learning. ICML 2009, Montreal, Quebec, Canada, June 14–18, 2009, In: ACM International Conference Proceeding Series, 382, ACM, pp. 529–536. http://dx.doi.org/10.1145/1553374.1553443.

de Lara, N., Pineau, E., 2018. A simple baseline algorithm for graph classification. CoRR abs/1810.09155 arXiv:1810.09155.

de Leoni, M., 2022. Foundations of process enhancement. In: van der Aalst, W.M., Carmona, J. (Eds.), Process Mining Handbook. In: Lecture Notes in Business Information Processing, vol. 448, Springer, pp. 243–273. http://dx.doi.org/10.1007/978-3-031-08848-3_8.

de Leoni, M., van der Aalst, W.M.P., Dees, M., 2016. A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. Inf. Syst. 56, 235–257. http://dx.doi.org/10.1016/j.is.2015.07.003.

Leontjeva, A., Conforti, R., Francescomarino, C.D., Dumas, M., Maggi, F.M., 2015. Complex symbolic sequence encodings for predictive monitoring of business processes. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (Eds.), Business Process Management - 13th International Conference, Proceedings. BPM 2015, Innsbruck, Austria, August 31 - September 3, 2015, In: Lecture Notes in Computer Science, vol. 9253, Springer, pp. 297–313. http://dx.doi.org/10.1007/978-3-319-23063-4_21.

Li, G., de Carvalho, R.M., van der Aalst, W.M.P., 2017. Automatic discovery of object-centric behavioral constraint models. In: Abramowicz, W. (Ed.), Business Information Systems - 20th International Conference, Proceedings. BIS 2017, June 28–30, Poznan, Poland, 2017, In: Lecture Notes in Business Information Processing, vol. 288, Springer, pp. 43–58. http://dx.doi.org/10.1007/978-3-319-59336-4_4.

Li, K., Gao, X., Jia, X., Xue, B., Fu, S., Liu, Z., Huang, X., Huang, Z., 2022. Detection of local and clustered outliers based on the density-distance decision graph. Eng. Appl. Artif. Intell. 110, 104719. http://dx.doi.org/10.1016/j.engappai.2022.104719.

Li, G., de Murillas, E.G.L., de Carvalho, R.M., van der Aalst, W.M.P., 2018. Extracting object-centric event logs to support process mining on databases. In: Mendling, J., Mouratidis, H. (Eds.), Information Systems in the Big Data Era - CAiSE Forum 2018, June 11–15, 2018, Proceedings. In: Lecture Notes in Business Information Processing, vol. 317, Springer, Tallinn, Estonia, pp. 182–199. http://dx.doi.org/10.1007/978-3-319-92901-9_16.

Lu, X., Nagelkerke, M., van de Wiel, D., Fahland, D., 2015. Discovering interacting artifacts from ERP systems. IEEE Trans. Serv. Comput. 8 (6), 861–873. http://dx.doi.org/10.1109/TSC.2015.2474358.

Ma, H., Bian, Y., Rong, Y., Huang, W., Xu, T., Xie, W., Ye, G., Huang, J., 2022. Cross-dependent graph neural networks for molecular property prediction. Bioinformatics 38 (7), 2003–2009. http://dx.doi.org/10.1093/bioinformatics/btac039.

Meyer, A., Pufahl, L., Fahland, D., Weske, M., 2013. Modeling and enacting complex data dependencies in business processes. In: Daniel, F., Wang, J., Weber, B. (Eds.), Business Process Management - 11th International Conference, Proceedings. BPM

2013, Beijing, China, August 26–30, 2013, In: Lecture Notes in Computer Science, vol. 8094, Springer, pp. 171–186. http://dx.doi.org/10.1007/978-3-642-40176-3_14.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J., 2013. Distributed representations of words and phrases and their compositionality. In: Burges, C.J.C., Bottou, L., Ghahramani, Z., Weinberger, K.Q. (Eds.), Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a Meeting Held December 5–8, 2013. Lake Tahoe, Nevada, United States, pp. 3111–3119.

Mishra, K., Basu, S., Maulik, U., 2022. *Graft*: A graph based time series data mining framework. Eng. Appl. Artif. Intell. 110, 104695. http://dx.doi.org/10.1016/j.engappai.2022.104695.

Moctar-M'Baba, L., Assy, N., Sellami, M., Gaaloul, W., Nanne, M.F., 2022. Extracting artifact-centric event logs from blockchain applications. In: Ardagna, C.A., Bian, H., Chang, C.K., Chang, R.N., Damiani, E., Dustdar, S., Marco, J., Singh, M.P., Teniente, E., Ward, R., Wang, Z., Xhafa, F., Zhang, J. (Eds.), IEEE International Conference on Services Computing. SCC 2022, Barcelona, Spain, July 10–16, 2022, IEEE, pp. 274–283. http://dx.doi.org/10.1109/SCC55611.2022.00048.

Montali, M., Rivkin, A., 2017. DB-nets: On the marriage of colored Petri nets and relational databases. Trans. PEtri Nets Other Model. Concurr. 12, 91–118. http://dx.doi.org/10.1007/978-3-662-55862-1_5.

de Murillas, E.G.L., Reijers, H.A., van der Aalst, W.M.P., 2020. Case notion discovery and recommendation: Automated event log building on databases. Knowl. Inf. Syst. 62 (7), 2539–2575. http://dx.doi.org/10.1007/s10115-019-01430-6.

Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., Jaiswal, S., 2017. graph2vec: Learning distributed representations of graphs. CoRR abs/1707.05005 arXiv:1707.05005.

Nooijen, E.H.J., van Dongen, B.F., Fahland, D., 2012. Automatic discovery of data-centric and artifact-centric processes. In: Rosa, M.L., Soffer, P. (Eds.), Business Process Management Workshops - International Workshops, Revised Papers. BPM 2012, Tallinn, Estonia, September 3, 2012, In: Lecture Notes in Business Information Processing, vol. 132, Springer, pp. 316–327. http://dx.doi.org/10.1007/978-3-642-36285-9_36.

Park, G., van der Aalst, W.M.P., 2022. Monitoring constraints in business processes using object-centric constraint graphs. CoRR abs/2210.12080 arXiv:2210.12080.

Park, G., Adams, J.N., van der Aalst, W.M.P., 2022. OPerA: Object-centric performance analysis. In: Ralyté, J., Chakravarthy, S., Mohania, M., Jeusfeld, M.A., Karlapalem, K. (Eds.), Conceptual Modeling - 41st International Conference, Proceedings. ER 2022, Hyderabad, India, October 17–20, 2022, In: Lecture Notes in Computer Science, vol. 13607, Springer, pp. 281–292. http://dx.doi.org/10.1007/978-3-031-17995-2_20.

Popova, V., Fahland, D., Dumas, M., 2015. Artifact lifecycle discovery. Int. J. Coop. Inf. Syst. 24 (1), 1550001:1–1550001:44. http://dx.doi.org/10.1142/S021884301550001X.

Rozemberczki, B., Kiss, O., Sarkar, R., 2020. Karate club: An API oriented open-source Python framework for unsupervised learning on graphs. In: d'Aquin, M., Dietze, S., Hauff, C., Curry, E., Cudré-Mauroux, P. (Eds.), The 29th ACM International Conference on Information and Knowledge Management, Virtual Event. CIKM '20, Ireland, October 19–23, 2020, ACM, pp. 3125–3132. http://dx.doi.org/10.1145/3340531.3412757.

Sani, M.F., van Zelst, S.J., van der Aalst, W.M.P., 2018. Applying sequence mining for outlier detection in process mining. In: Panetto, H., Debruyne, C., Proper, H.A., Ardagna, C.A., Roman, D., Meersman, R. (Eds.), On the Move to Meaningful Internet Systems. OTM 2018 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2018, October 22–26, 2018, Proceedings, Part II. In: Lecture Notes in Computer Science, vol. 11230, Springer, Valletta, Malta, pp. 98–116. http://dx.doi.org/10.1007/978-3-030-02671-4_6.

Shervashidze, N., Schweitzer, P., van Leeuwen, E.J., Mehlhorn, K., Borgwardt, K.M., 2011. Weisfeiler–Lehman graph kernels. J. Mach. Learn. Res. 12, 2539–2561. http://dx.doi.org/10.5555/1953048.2078187.

Tax, N., Verenich, I., Rosa, M.L., Dumas, M., 2017. Predictive business process monitoring with LSTM neural networks. In: Dubois, E., Pohl, K. (Eds.), Advanced Information Systems Engineering - 29th International Conference, Proceedings. CAiSE 2017, Essen, Germany, June 12–16, 2017, In: Lecture Notes in Computer Science, vol. 10253, Springer, pp. 477–492. http://dx.doi.org/10.1007/978-3-319-59536-8_30.

Teinemaa, I., Dumas, M., Rosa, M.L., Maggi, F.M., 2019. Outcome-oriented predictive process monitoring: Review and benchmark. ACM Trans. Knowl. Discov. Data 13 (2), 17:1–17:57. http://dx.doi.org/10.1145/3301300.

Tsitsulin, A., Mottin, D., Karras, P., Bronstein, A.M., Müller, E., 2018. NetLSD: Hearing the shape of a graph. In: Guo, Y., Farooq, F. (Eds.), Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. KDD 2018, London, UK, August 19–23, 2018, ACM, pp. 2347–2356. http://dx.doi.org/10.1145/3219819.3219991.

Verma, S., Zhang, Z., 2017. Hunt for the unique, stable, sparse and fast feature learning on graphs. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (Eds.), Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4–9, 2017. Long Beach, CA, USA, pp. 88–98.

Vishwanathan, S.V.N., Schraudolph, N.N., Kondor, R., Borgwardt, K.M., 2010. Graph kernels. J. Mach. Learn. Res. 11, 1201–1242.

Waibel, P., Pfahlsberger, L., Revoredo, K., Mendling, J., 2022. Causal process mining from relational databases with domain knowledge. CoRR abs/2202.08314 arXiv:2202.08314.

Wang, L., Huang, C., Ma, W., Cao, X., Vosoughi, S., 2021. Graph embedding via diffusion-wavelets-based node feature distribution characterization. In: Demartini, G., Zuccon, G., Culpepper, J.S., Huang, Z., Tong, H. (Eds.), The 30th ACM International Conference on Information and Knowledge Management, Virtual Event. CIKM '21, Queensland, Australia, November 1 - 5, 2021, ACM, pp. 3478–3482. http://dx.doi.org/10.1145/3459637.3482115.

Weber, I., Farshchi, M., Mendling, J., Schneider, J., 2015. Mining processes with multi-instantiation. In: Wainwright, R.L., Corchado, J.M., Bechini, A., Hong, J. (Eds.), Proceedings of the 30th Annual ACM Symposium on Applied Computing, April 13–17, 2015. ACM, Salamanca, Spain, pp. 1231–1237. http://dx.doi.org/10.1145/2695664.2699493.

Weerdt, J.D., Wynn, M.T., 2022. Foundations of process event data. In: van der Aalst, W.M.P., Carmona, J. (Eds.), Process Mining Handbook. In: Lecture Notes in Business Information Processing, vol. 448, Springer, pp. 193–211. http://dx.doi.org/10.1007/978-3-031-08848-3_6.

van der Werf, J.M.E.M., Polyvyanyy, A., 2020. The information systems modeling suite - modeling the interplay between information and processes. In: Janicki, R., Sidorova, N., Chatain, T. (Eds.), Application and Theory of Petri Nets and Concurrency - 41st International Conference, Proceedings. PETRI NETS 2020, Paris, France, June 24–25, 2020, In: Lecture Notes in Computer Science, vol. 12152, Springer, pp. 414–425. http://dx.doi.org/10.1007/978-3-030-51831-8_22.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S., 2021. A comprehensive survey on graph neural networks. IEEE Trans. Neural Netw. Learn. Syst. 32 (1), 4–24. http://dx.doi.org/10.1109/TNNLS.2020.2978386.

Yanardag, P., Vishwanathan, S.V.N., 2015. Deep graph kernels. In: Cao, L., Zhang, C., Joachims, T., Webb, G.I., Margineantu, D.D., Williams, G. (Eds.), Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 10-13, 2015. ACM, Sydney, NSW, Australia, pp. 1365–1374. http://dx.doi.org/10.1145/2783258.2783417.

Yang, H., Park, M., Cho, M., Song, M., Kim, S., 2014. A system architecture for manufacturing process analysis based on big data and process mining techniques. In: Lin, J., Pei, J., Hu, X., Chang, W., Nambiar, R., Aggarwal, C.C., Cercone, N., Honavar, V.G., Huan, J., Mobasher, B., Pyne, S. (Eds.), 2014 IEEE International Conference on Big Data. IEEE BigData 2014, Washington, DC, USA, October 27–30, 2014, IEEE Computer Society, pp. 1024–1029. http://dx.doi.org/10.1109/BigData.2014.7004336.

Zandkarimi, F., Rehse, J., Soudmand, P., Hoehle, H., 2020. A generic framework for trace clustering in process mining. In: van Dongen, B.F., Montali, M., Wynn, M.T. (Eds.), 2nd International Conference on Process Mining. ICPM 2020, Padua, Italy, October 4–9, 2020, IEEE, pp. 177–184. http://dx.doi.org/10.1109/ICPM49681.2020.00034.

Zhang, Q., Yin, C., Chen, Y., Su, F., 2022. IGCRRN: improved graph convolution res-recurrent network for spatio-temporal dependence capturing and traffic flow prediction. Eng. Appl. Artif. Intell. 114, 105179. http://dx.doi.org/10.1016/j.engappai.2022.105179.