Global Predictive Monitoring of Object-Centric Processes

Massimiliano de Leoni, Pietro Volpato

Department of Mathematics
University of Padua, Italy
deleoni@math.unipd.it,
pietro.volpato.2@studenti.unipd.it

Abstract. Object-centric processes follow a paradigm in which a single process instance does not operate in isolation but interacts with other instances of the same or different processes. Recently, these processes have become increasingly popular in both academia and industry due to their relevance in various application scenarios. Predictive process monitoring is naturally still relevant for object-centric processes to predict the final outcome of individual process executions. A substantial body of research work exists to tackle the predictive monitoring task for object-centric processes. However, while object-centric processes already consider the interactions among objects, there may still be interferences among executions that are not explicitly represented through shared objects. Existing techniques for predictive monitoring for object-centric processes only consider the explicit object interactions, overlooking the hidden interferences. If these interferences have a significant influence on predicted KPIs of interest, the prediction accuracy is negatively impacted. This paper puts forward a technique that performs global predictions of all ongoing process executions together. Experiments on multiple processes show that indeed these global predictions are significantly more accurate, compared to the setting in which process executions are predicted in isolation by only considering the explicit interactions.

Keywords: Object-centric Processes · Predictive Monitoring · Graph Neural Networks

1 Introduction

Predictive process monitoring aims to anticipate the future state of ongoing process executions, using historical event data [6]. This type of monitoring is aimed at forecasting key aspects of a process, such as its completion time, the customer satisfaction, or the likelihood of achieving a desired outcome (e.g., success or failure). The goal is to anticipate issues of running process executions, and thus reason on corrective actions before the consequences can no longer be mitigated.

Predictive process monitoring has traditionally assumed that executions follow a sequence-like flow where each execution is associated with data objects that are not shared among different executions [6]. In reality, the process executions are much

more complex: activities can be related to multiple objects of different types simultaneously [7], which are shared among activities of different executions. These considerations led to the introduction of the paradigm of object-centric processes. Predictive process monitoring is naturally still relevant for object-centric processes. This broader assumption implies that each activity execution can have multiple predecessors and successors in different executions, breaking the assumption of a linear sequence of process executions, as assumed in traditional predictive process monitoring.

The last couple of years have witnessed the introduction of several techniques to address object-centric predictive process monitoring (cf. Section 2), where graph-like flow process representations have replaced sequence-like flows. However, while predictive monitoring for object-centric processes already considers the explicit interactions between objects, there may be interference factors among process executions that are not explicitly encoded through the object interactions, and thus are ignored when performing predictions. Consider a global supply chain for electronics. Each shipment is a process execution that involves multiple stages (order processing, warehousing, transportation, and delivery). A shipment's delivery time (prediction target) is not solely determined by its individual route, and by the number of items and trucks, which can be shared among process executions. It is also influenced by other factors, such as the routing of other shipments, sudden spikes in the number of shipments, multiple shipments requiring the same specialized transport (e.g., refrigerated trucks). These factors can cause bottlenecks and delays. Some of these factors may not be encoded through object interactions; for instance, trucks might not be process' objects, or shipments might not be explicitly linked together. The existing body of research on predictive monitoring for object-centric processes would ignore these interactions, thereby potentially negatively affecting the quality of the predictions.

This work aims to provide a new structural framework for encoding process executions, where graphs of the ongoing process executions are merged into a single graph, thus enabling predictors to be globally trained, namely training them all together to predict all KPI values in one shot. Our framework facilitates information flow within the graph, enhancing the predictive performance of a unified representation compared to treating process executions independently.

Our framework leverages on Graph Neural Networks to perform the predictions, because process executions are naturally organized in form of graphs. A full-fledged prototype has been developed in Python, and tested on multiple object-centric processes and KPIs.[1] The experimental results show that global predictive process monitoring for object-centric processes leads to a higher prediction accuracy, when compared with the state-of-the-art techniques, which ignore the hidden interferences among process executions.

The remainder of the paper is organized as follows. First, we discuss related works in Section 2, followed by the necessary preliminaries in Section 3. Section 4 formalizes the problem of object-centric process prediction, while Section 5 introduces our theoretical framework to address the prediction problem. Section 6 discusses the experimental setup and the results of the assessment. Section 7 discusses the current limitations and

[1] `https://figshare.com/s/60b996e372eebe097517`

the consequent avenues for future work, while Section 8 concludes the paper, summarizing and discussing the contributions.

2 Related Works

Research interest in Predictive Process Monitoring (PPM) within the object-centric paradigm has grown significantly in recent years. Early approaches to solving the prediction problem flattened the Object-Centric Event Log (OCEL) into a traditional format. This approach, despite being simple, results in a loss of structural information, essential for preserving the object-centric nature of the process to analyze.

Both Galanti et al. [8] and Gherissi et al. [9] followed this idea, mitigating the effects of flattening through feature enhancement. The added features relate to information about the number of objects, their aggregated attributes and the percentage of objects that have undergone a specific activity. Flattening is also associated with the problems of deficiency, convergence and divergence, introduced in [1]. At the same time, other works have proposed native object-centric approaches to overcome flattening issues, exploiting the process' intrinsic nature.

In [5], Adams et al. have proposed to move away from the traditional case notion, introducing the concept of process execution to represent the event graphs of multiple interdependent objects. In this way, the relationship between objects and the dependencies between events is maintained. The authors also proposed a leading type extraction method, to generate multiple sub-graphs from connected components, to better handle complex and interconnected logs.

Building on this concept, Adams et al. proposed a graph-based encoding to retain the structural information of the process [3,2]. Their findings demonstrated that a native graph-based encoding outperforms a non-native tabular and sequential encoding. This new structure avoids the need for flattening and better represents dependencies in the process. The work in [4] showed how structural information alone, without the addition of extra event features, can outperform the current flattened state of the art approaches. Results were achieved using both GNN and graph embeddings.

All the approaches introduced so far either require flattening event logs or work with homogeneous graphs, which require aggregating object features and a consequent information loss. To address these limitations, Smit et al. [14] propose a heterogeneous graph structure, i.e., graphs with multiple node and edge types, with each object instance being represented by a unique node. This structure is much more representative than the homogeneous one, thereby yielding improvement over the current state of the art.

The works mentioned above and the rest of the state of the art on predictive monitoring for object-centric processes only consider the modeled interaction among objects. Therefore, they ignore the hidden interferences that are not explicitly represented through object interactions. In the remainder, we introduce our framework for global predictive monitoring of object-centric processes that introduces a graph encoding that integrates all concurrent process executions within a single structure. This encoding works globally and enables performing the predictions of the process executions collectively.

Table 1. Object-centric Event Log with Associated Objects

ev_id	Event Type	Timestamp	Order	Item	Package	Product	Customer	Employee
ev_19	Place Order	2023-04-05 04:24:56	{o6}					{e6}
ev_20	Confirm Order	2023-04-05 06:24:56	{o6}	{i16, i17}		{pr1, pr1}	{c1}	{e1}
ev_21	Confirm Order	2023-04-05 10:14:43	{o3}	{i5, i6, i7}		{pr2, pr3, pr4}	{c2}	{e1}
ev_22	Item Out Of Stock	2023-04-05 12:44:58		{i9}		{pr8}		{e2}
ev_23	Item Out Of Stock	2023-04-05 12:44:58		{i15}		{pr8}		{e3}
ev_24	Pick Item	2023-04-05 13:04:56		{i16}		{pr1}		{e2}
ev_25	Pick Item	2023-04-05 13:04:56		{i17}		{pr1}		{e2}
ev_26	Place Order	2023-04-05 13:05:45	{o7}	{i18, i19, i20}		{pr10, pr6, pr11}	{c3}	
ev_27	Payment Reminder	2023-04-05 15:31:38	{o1}	{i1, i2, i3}		{pr9, pr6, pr7}		
ev_28	Pay Order	2023-04-05 17:13:35	{o2}	{i4}		{pr5}		
ev_29	Create Package	2023-04-05 17:31:49		{i16, i17}	{p1}	{pr1, pr1}		{e4}
ev_30	Send Package	2023-04-05 23:31:49			{p1}			{e5}
ev_31	Package Delivery	2023-04-08 14:30:49			{p1}			{e6}

Our framework does not require flattening, nor does it require an explicit feature aggregation, preserving the inherent complexity of the object-centric process executions to analyze. We achieve this through the use of a heterogeneous graph structure, better suited for capturing inter-dependencies and relationships between different objects and events in the process.

3 Preliminaries

This section introduces key concepts for manipulating object-centric event data, adapting the definitions from [5] to align with the OCEL 2.0 standard and our framework. We also formalize the graph prediction problem, to fit heterogeneous graph neural networks to our proposed framework.

The formalization of Object-centric Event Log (OCEL) requires the introduction of the following universes: the universe of events is denoted by U_{ev}, the universe of activities (i.e. event types) is denoted by U_{etype} and the universe of timestamps by U_{time}. U_{obj} is the universe of objects, with U_{otype} being the universe of object types. Events and objects both have attributes having values, with the universe of attributes being U_{attr} and the universe of attribute values being U_{val}. The universe of qualifiers describing relationships among entities is defined as U_{qual}.

Definition 1 (Object-Centric Event Log). *An* Object-Centric Event Log 2.0 (OCEL) *is a tuple* $L = (E, O, EA, OA, evtype, time, objtype, eaval, oaval, \pi_{trace}, eatype, oatype, E2O, O2O)$ *where:*

- $E \subseteq U_{ev}$ *is the set of events.*
- $O \subseteq U_{obj}$ *is the set of objects.*
- $EA \subseteq U_{attr}$ *is the set of event attributes.*
- $OA \subseteq U_{attr}$ *is the set of object attributes.*
- $evtype : E \rightarrow U_{etype}$ *assigns types to events.*
- $time : E \rightarrow U_{time}$ *assigns timestamps to events.*
- $objtype : O \rightarrow U_{otype}$ *assigns types to objects.*

- *eaval* : $(E \times EA) \nrightarrow U_{val}$ *assigns values to event attributes (partial function).*
- *oaval* : $(O \times OA \times U_{time}) \nrightarrow U_{val}$ *assigns values to object attributes over time (partial function).*
- $\pi_{trace} : O \rightarrow E^*$ *maps each object to a time-ordered sequence of related events, such that for every object* $o \in O$, $\pi_{trace}(o) = \langle e_1, \ldots, e_n \rangle$ *where* $\forall i \in \{1, \ldots, n-1\}$. *time*$(e_i) \leq$ *time*(e_{i+1}) *and* $\forall i \in \{1, \ldots, n\}, \exists s \in U_{qual} : (e_i, s, o) \in E2O$
- *eatype* : $EA \rightarrow U_{etype}$ *assigns types to event attributes.*
- *oatype* : $OA \rightarrow U_{otype}$ *assigns types to object attributes.*
- $E2O \subseteq E \times U_{qual} \times O$ *represents qualified event-to-object relations.*
- $O2O \subseteq O \times U_{qual} \times O$ *represents qualified object-to-object relations.*

Example 1. *Table 1 presents a visual representation of an object-centric event log that conforms to Definition 1. The set of events is given by* $E = \{ev\_19, ev\_21, \ldots, ev\_31\}$, *and the set of objects by* $O = \{o1, o2, \ldots, o6, i16, \ldots, e6\}$. *As examples, the pairs* $(ev\_20, o6)$ *and* $(ev\_20, i16)$ *belong to the event-to-object relation* $E2O$, *since ev_20 is associated with both o6 and i16 (see the first event in Table 1). Additional examples include: objtype*$(o6) =$ *Order,* $\pi_{trace}(i16) = \langle ev\_20, ev\_24, ev\_29 \rangle$, *and time*$(ev\_20) =$ *2023-04-05 06:24:56. Furthermore, suppose that ev_20 assigns the value* 2 *to the event attribute* n, *i.e., eaval*$(ev\_20, n) = 2$. *If value two is used to update the attribute num_items of object o6 at the timestamp of ev_20, then oaval*$(o6,$ *num_items*, *2023-04-05 06:24:56*$) = 2$.
Note that the object-to-object relation $O2O$ *cannot be directly derived from Table 1. For instance, the pair* $(o6, i16)$ *is likely included in* $O2O$ *because these objects co-occur in ev_20. However, in general,* $O2O$ *is not simply the set of object pairs that co-occur in events. Some object-to-object relations may originate from domain-specific links that go beyond shared event participation.*

Before defining process executions, we must establish how entities interact in an object-centric process. By leveraging the relational tables in event logs, we can effectively model the relationships among objects and their interactions with events.

Definition 2 (Event-Object Graph). *Let* $L = (E, O, EA, OA,$ *evtype*, *time*, *objtype*, *eaval*, *oaval*, π_{trace}, *eatype*, *oatype*, $E2O, O2O)$ *be an object-centric event log. The event-object graph is an undirected graph* $OG_L = (O \cup E, C_{OE})$ *where the set of undirected edges is defined as* $C_{OE} = \{\{x, y\} \mid (x, y) \in E2O \cup O2O \vee (y, x) \in E2O \cup O2O$.

We denote **the distance of two nodes** n_1, n_2 within OG_L with the function $dist : O \cup E \times O \cup E \rightarrow \mathbb{N}_0$, where, for each pairs o_1, o_2 of nodes, $dist(o_1, o_2 = \infty$ if the nodes are not connected. To represent the complexity of the new event-object graph, we need a structure that can handle different node and relationship types. This can be achieved by using heterogeneous graphs.

In line with Adams et al. [5], we introduce the concept of viewpoint as a chosen leading object type, which induces the executions of an object-centric process:[2]

Definition 3 (Process Executions). *Let* $L = (E, O, EA, OA,$ *evtype*, *time*, *objtype*, *eaval*, *oaval*, π_{trace}, *eatype*, *oatype*, $E2O, O2O)$ *be an event log, let* $OG_L = (O \cup E, C_{OE})$ *be its event-object graph, let* $\mathcal{V} \in U_{otype}$ *be a viewpoint. For each object* $o \in O$ *of* $objtype(o) = \mathcal{V}$, *a process execution for* o *is an undirected graph* (N, A)

[2] Given two sequences σ, σ', $\sigma' \subseteq \sigma$ indicates that σ' is a sub-sequence of σ, with possibly being $\sigma = \sigma'$.

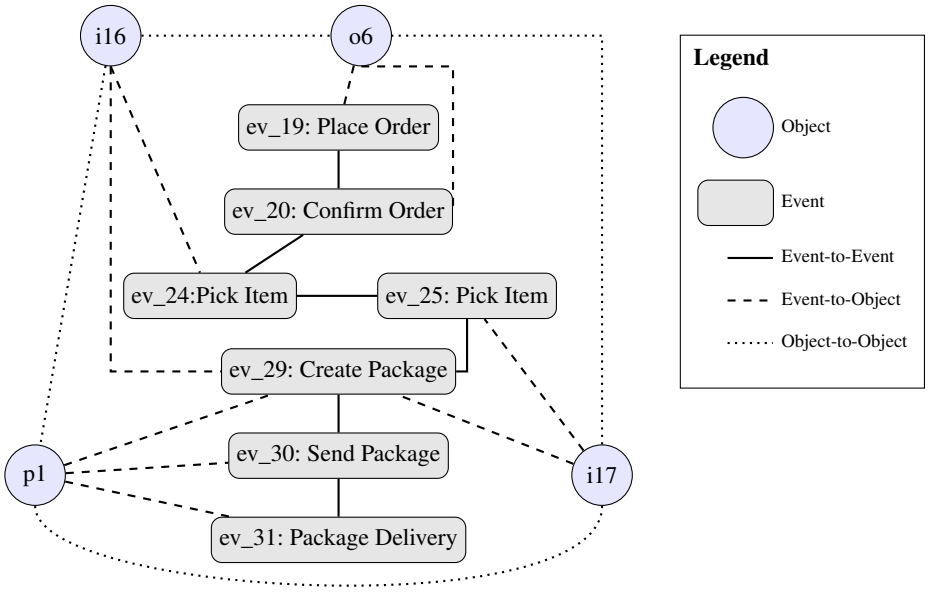

Fig. 1. Example of a process execution extracted from the object-centric event log in Table 1 for order *o6*. Objects for products, customers and employees are not shown to ensure readability.

where $N = O' \cup E'$ where O' is the set of objects assigned to o and E' is the set of events related to objects in O':

- $O' = \{o'' \in O \mid \textit{objtype}(o'') \neq \mathcal{V} \wedge \forall o' \in O.\ \textit{objtype}(o') = \mathcal{V} \Rightarrow \textit{dist}(o', o'') \geq \textit{dist}(o, o'')\}$
- $E' = \{e \in E \mid \forall o' \in O.\ \textit{objtype}(o') = \mathcal{V} \Rightarrow \textit{dist}(e, o') \geq \textit{dist}(e, o)\}$.

Arcs are thus as follows: $A = \{\{e, \top, e'\} \mid \exists o \in O'.\langle e, e'\rangle \subseteq \pi_{trace}(o)\} \cup \{\{a, s, b\} \in C_{OE} \mid a \in N, b \in N\}$.

Note that the definition above introduces a fictitious $\top$ to qualify the arcs between events, assuming $\top \in U_{qual}$ without losing generality. Figure 1 illustrates a process execution, where an order contains two items assigned to the same package. The legend distinguishes three types of relationships among entities: two directly derived from the event log and one generated in the preliminaries.

The use of a heterogeneous structure to represent process executions eliminates the need for feature aggregation. Each object and event within an execution can be directly associated with a unique node, preserving its individual characteristics. In contrast, previous homogeneous approaches where nodes represented only events, required condensing information about interacting objects into the event representation. This not only introduced an additional preprocessing step, but also led to a less meaningful representation of the process to analyze.

4 Object-centric Process Prediction

This section formalizes the problem and reviews state-of-the-art techniques. First, we define how key performance indicators (KPIs) are computed (cf. Definition 4), and formalize the object-prediction problem (see Section 4.1). Finally, Section 4.2 formalizes how the object-centric prediction problem is currently tackled by the literature.

Definition 4 (KPI). *Let P be a set of process executions. A KPI is a function $K : P \times U_{time} \to \mathbb{R}^n$ that, given any process execution $p \in P$, returns a vector $K(p,t)$ of the KPI values at time t.*

One advantage of using a heterogeneous structure to represent process executions is the presence of multiple node types. In our framework, each node type is associated with an object type, so in the same process execution, we have distinct entities that describe different objects. By applying the K function iteratively to the same process execution, we can obtain different metrics related to different object types.

4.1 Formalization of the Problem

The concept of process execution introduced so far considers the latter to be completed. This is essential for computing the KPI of reference but presents a limitation at inference time. Our goal is to predict KPIs even when the process execution is at an intermediate state, allowing for real-time monitoring and early decision-making.

A formal definition of the object-centric process prediction problem requires the introduction of the concept of a prefix of a process execution:

Definition 5 (Prefix of a Process Execution). *Let $L = (E, O, EA, OA,$ $evtype, time, objtype, eaval, oaval, \pi_{trace}, eatype, oatype, E2O, O2O)$ be an event log. Let P be the universe of potential process executions from L. The function $Prefix : P \times U_{time} \to P$ returns the snapshot of process executions: given a timestamp t and a process execution $p \in P$, $Prefix(p,t)$ returns the snapshot of p at time t. Formally, given a process execution $p = (N, A)$, $Prefix(p,t) = (N', A')$ where $N' = E' \cup O'$ with:*

- $E' = \{e \in E \cap N \mid time(e) \leq t\}$
- $O' = \{o \in O \cap N \mid \exists e \in \pi_{trace}(o).time(e) \leq t\}$.
- $A' = \{(a, s, b) \in A \mid a \in N', b \in N'\}$

The definition of prefix allows us, given a timestamp, to include in it only objects and events that have already taken place, excluding future behaviors. Note that a prefix $\overline{p}$ of a process execution p is still a process execution, which is a subgraph of p, at any time. With this concept at hand, we can formalize the problem of object-centric process prediction as follows:

Definition 6 (Object-centric Process Prediction). *Let P be a complete process execution, and let P' be its prefix observed at time t, i.e., $P' = Prefix(P,t)$. The goal of Object-centric Process Prediction is to compute $K(P',t)$ where the computation of $K(P',t)$ may depend on the full execution P beyond t. Therefore, computing $K(P',t)$ requires estimating the key elements of the structure of P.*

The following are examples of sensible KPIs related for the object-centric event log in Table 1:

Example 2. *Let us again consider the object type Order as viewpoint. Let p_2 and p_6 be the process execution related to orders o2 and o6 in Table 1. Suppose that the execution is still at time $\bar{t} = 2024-04-01\ 12:00:00$ when only prefixes $p'_2 = Prefix(p_2, \bar{t})$ and $p'_6 = Prefix(p_6, \bar{t})$. The following are computation examples of three potential KPIs computed on p'_2 or p'_6:*

Time until an order is paid. *For order o2, $K_{order}(p'_2, t) =$ (2023-04-05 17:13:35) $- t = [364415]$ seconds. Note that (2023-04-05 17:13:35) is when the order o2 is paid, which only happens later than $\bar{t}$.*

Time until items are packaged. *The process execution p_6 related to o6 contains an event for Create Package at time 2023-04-05 17:31:49 for two items i16 and i17 (event ev_29 in Table 1). Therefore, the time until these items are packaged is (2023-04-05 17:31:49) $- t = 3655509$ seconds. Therefore, $K_{packaged}(p'_6, t) = [365509, 365509]$. The vector contains two elements because a package is created for the two items i16 and i17, and the values are the same because packaged together.*

Average Number of item per package. *Following the reasoning of the KPI, $K_{item}(p'_6, t) = [2]$. Note that the computation of $K_{item}(p', t)$ is again based on the complete process execution p, which is unknown at time t because only prefix p' has been observed at time t.*

4.2 Training of a Graph-based Object-Centric Process Predictor

This section discusses object-centric process prediction techniques from the literature that preserve the graph structure of process executions, rather than applying event-log flattening. These approaches, previously introduced in Section 2, maintain the native object-centric representation by modeling process executions as graphs. Such techniques typically employ graph-based predictive models, such as graph neural networks, which are designed to directly operate on graph-structured data, such as object-centric process executions.

Formally, let G denote the universe of valid input graphs. A graph-based predictor defines a function $\mathcal{F} : G \to \mathbb{R}^n$ that maps an input graph $g \in G$ to a vector of predicted values corresponding to n response variables. The model is trained on a multiset of labeled examples drawn from $G \times \mathbb{R}^n$, commonly referred to as the learning set in the literature.

In the context of object-centric process prediction, given a KPI $K : P \times U_{time} \to \mathbb{R}^n$, for each process execution p and relevant time t, a learning sample $(Prefix(p,t), K(Prefix(p,t),t)$ is built. At training time, it is possible to compute $K(\text{Prefix}(p,t),t)$, since the full evolution of $\text{Prefix}(p,t)$ within the training dataset is known, including how it leads to the complete process execution p. .

Definition 7 (Learning set). *Let $L = (E, O, EA, OA, evtype, time, objtype, eaval, oaval, \pi_{trace}, eatype, oatype, E2O, O2O)$ be an event log. Let P be the universe of potential executions for the process to which L refers. Let $P_{learning} = \{(N_1, A_1), \ldots, (N_m, A_m)\} \subset P$ be the process executions in L. Let K: $P \times U_{time} \to \mathbb{R}^n$ be a KPI function of interest. Let $\mathcal{T} = \bigcup_{i=1}^{m} \bigcup_{e \in (N_i \cap E)} time(e)$*

be the set of the relevant timestamps for $P_{learning}$. The learning set is built as follows:

$$\bigcup_{p \in P_{learning}} \bigcup_{t \in \mathcal{T}} \Big(Prefix(p,t), K(Prefix(p,t),t) \Big)$$

Example 3. *To illustrate, consider two orders o_1 and o_2, each comprising a single item i_1 and i_2, respectively. Both items are shipped in the same package p_1. When item i_1 is ready for shipment, the subgraph $\overline{p}_{o_1}$ remains unchanged, with the subgraph related to $\overline{p}_{o_2}$ evolving until i_2 is ready to be shipped. At that point, both subgraphs will grow until the package is shipped and orders can be considered fulfilled.*

5 Framework

The execution of a process instance in the processes under consideration is strongly influenced by other concurrently running cases. Treating each execution in isolation would negatively impact the performance of predictive models, since we would be omitting critical contextual information. This limitation can be addressed by using a structure that inherently captures these dependencies, such as a graph representation that encodes all the concurrently running executions.

5.1 Global Training in Predictive Monitoring

In our technique, every active process execution is merged into a single heterogeneous graph, which is used as a training instance for graph-based predictors. This ensures that the hidden interferences among process executions are considered when the predictor is trained. This approach mirrors real-world scenarios, where at inference time a company may have *n* active executions, and wants to compute metrics on the their most updated representation. When the training instances are merged, the response variables, namely the KPIs, are also merged:[3]

Definition 8 (Merged Learning Set). *Let $L = (E, O, EA, OA, evtype, time, objtype, eaval, oaval, \pi_{trace}, eatype, oatype, E2O, O2O)$ be an event log. Let P be the universe of potential executions for the process to which L refers. Let $P_{learning} = \{(N_1, A_1), \ldots, (N_m, A_m)\} \subset P$ be the process executions in L. Let $K : P \times U_{time} \to \mathbb{R}^n$ be a KPI function of interest. Let $\mathcal{T} = \bigcup_{i=1}^{m} \bigcup_{e \in (N_i \cap E)} time(e)$ be the set of the relevant timestamps for $P_{learning}$. The training set is built as follows:*

$$\bigcup_{t \in \mathcal{T}} \Big(\bigcup_{p \in P_{learning}} Prefix(p,t), \bigoplus_{p \in P_{learning}} K(Prefix(p,t),t) \Big)$$

5.2 Feature Encoding

When used as input for a GNN, a process execution *p* must be transformed into a suitable format. A distinct node type is created for each object type in *p*, and one additional node type is used to represent event nodes. The information is encoded as follows:

[3] Let $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, \ldots, y_m)$ be any two vectors. Their concatenation is denoted by $\boldsymbol{x} \oplus \boldsymbol{y} = (x_1, \ldots, x_n, y_1, \ldots, y_m)$.

Table 2. Characteristics of object-centric processes used in our experiments.

Process	Events	Activities	Object Instances	Object Types	Viewpoint	Correlation Factor	Avg Size of Process Executions
Order Management	20909	11	10917	6	Order	No. active orders	50
Logistics	121839	14	47543	7	Customer Order	No. active containers	121
IoT	9967	14	3956	5	Pickup Plan	No. active trucks	74

- **Event nodes**: Events are treated as categorical variables, where the set of event types is used as domain. Each event node is represented using a one-hot encoding of its respective event type.
- **Object nodes**: Object nodes only contain the raw features present in the event log, without further preprocessing. Categorical attributes in the event log are encoded using one-hot encoding.

Edges do not carry attributes; instead, they are represented using a binary indicator denoting the existence of a connection between two nodes.

To standardize the response variables (i.e., the KPIs), we used z-score normalization, also known as standardization, is a data preprocessing technique that transforms data to have a mean of 0 and a standard deviation of 1 [10]. This is achieved by subtracting the mean of the feature from each data point and then dividing by the standard deviation of the response variable of the training set. Z-normalization facilitates faster convergence in gradient descent-based algorithms, such as those used in training Graph Neural Networks, helps in outlier detection, and mitigates the influence of features with large numerical ranges [10].

6 Experiments

This section discusses the experimental setup and results. In detail, Section 6.1 introduces the processes, OCELs and KPIs used in the experiments, while Section 6.2 reports on the implemented GNN architecture and its respective hyperparameters. Section 6.3 reports on the evaluation methodology, where Section 6.4 reports and discusses the results. Finally, Section 6.5 discusses the asymptotic time complexity.

6.1 Processes, Event Logs and KPIs used in the Experiments

The datasets used in this work are three synthetic processes that are publicly available. The first is related to the process for the management of customer orders within a company, covering the steps from administration to shipping [11]. The second focuses on the logistics process behind a company selling goods overseas, from order registration to container management and shipping [11]. The third is related to a cargo pickup process where the moving goods are monitored via embedded IoT technologies [16]. The salient characteristics of the object-centric processes and logs are provided in the first four columns of Table 2, namely the number of events, activities, and object instances and types.

Table 3. The KPIs used in the experiment for each process, alongside the object type referring to the KPI of interest.

Process	KPI to predict	Object Type
Order Management	Time until Order is fulfilled	Order
	Time until the Item is delivered	Item
	Time until the Package is delivered	Package
Logistics	Time until the second to last event in the process execution	Order
	Time until the TD is fulfilled	Transport Document
	Time until Container is ready to depart	Container
IoT	Time until the Pickup Plan is fulfilled	Pickup Plan

The choice fell on these three case studies because they are the only publicly available case studies that are provided with a simulation model, which enables generating new OCELs. These models can be inspected and downloaded from the respective references, and are designed in CPN Tools.[4] As a matter of fact, none of the publicly available OCELs exhibits interactions with and interferences among the process executions. Of course, this does not imply that process executions cannot interfere with each other: given that every available OCEL is synthetically generated, the authors of the simulation models have likely not introduced these interferences because it was not their interest. Therefore, having a simulation model of object-centric processes is crucial in our experimental setting. It allows us to introduce interferences and assess their impact. Consequently, this enables us to evaluate whether our framework improves the accuracy of object-centric process predictions across scenarios with different levels of interference.

Specifically, for each process p and p's activity a, we started by considering the average m of the duration of a as defined in the CPN-Tools model from the literature. Then, the duration of activity a was sampled according to a discrete distribution with extremes $\theta \cdot (1-\alpha) \cdot m$ and $\theta \cdot (1+\alpha) \cdot m$ where α and θ are the noise and correlation factors, respectively. Specifically, we tested $\alpha = [0, 0.1, 0.2]$ and the correlation factor was dependent on the specific process being tested. The dedicated column in Table 2 details which correlation factor was chosen for each process. For instance, the duration of each activity of the process *Order Management* was given a correlation factor θ equal to the number of orders that were active at the moment when the activity was performed.

For order management and logistics processes, we defined and tested three KPIs. In contrast, only a single KPI was used for the cargo process with IoT devices (see Table 3). This distinction was necessary due to the nature of the dataset, where other objects, such as trucks or silos, were more peripheral and not meaningful for predicting the specific KPI.

For each object-centric process and event log, we used the same viewpoint, which is indicated in column *Viewpoint* in Table 2. Each viewpoint induces a set of process executions, and its average size is also provided in Table 2 (see last column). The size of a process execution is here measured as the sum of the number of nodes of the respective graph.

[4] CPN-Tools Web site – `https://cpntools.org/`

Table 4. Hyperparameter settings

Hyperparameter	Value(s)
Number of epochs	50
Patience	5
Number of message-passing layers	{2, 3, 4, 5}
Activation function	ReLU
Number of heads	{1, 3}
Optimizer	Adam
Number of hidden dimensions	{4, 6, 8, 10, 12}

6.2 The GNN Architecture and Hyperparameters

Our implementation builds on Graph Attention Layers [15]. The key innovation of Graph Attention Layers is the incorporation of an attention mechanism that assigns different weights to different neighbors of a node. These weights, learned dynamically from the data, enable the model to focus on the most relevant neighboring nodes.

The architecture also includes a pre-message-passing layer and a post-message-passing layer. The first layer processes the features before message passing, projecting them into a common embedding space. The final layer transforms the node embeddings, obtained after message passing, into the predicted KPI values.

A decreasing learning rate strategy was employed to train our models, with the values in the set $\{0.01, 0.0075, 0.005, 0.0025, 0.001, 0.0005\}$. This schedule allows for a higher learning rate in the initial epochs to accelerate convergence, followed by gradual reductions to refine the model's performance and stabilize training. An early stopping mechanism with a patience of 5 was employed to avoid overfitting. For the last process, we employed an exponential moving average of the validation loss, with an α value that started from 0.9 and was then decreased by 0.1 at each epoch.

6.3 Evaluation Methodology

To compare the performance difference between the two structures proposed in our framework, we followed the procedure outlined below. For each process, we used the viewpoint in Table 2, which defined the set of all process executions, as described in Definition 3. These process executions were then split into a training, a validation and a test process-execution set, with a proportion of 70%, 10% and 20%, respectively. Split was based on timestamps so that the process executions in the test process-execution set were temporally after those in the validation set, which in turn were after those in the training set. We included in the test sets only those process executions that started after the completion of the last process execution in the training and validation set. For each process execution in the training, validation and test process-execution sets, we generated the learning set and the merged learning set for training, validation and test. Learning sets were used to train and test the state-of-the-art techniques, whereas merged learning sets were employed to train and test our framework for global predictive monitoring of object-centric processes.

Every KPI used in the evaluation represents the remaining time until a specific condition c becomes true in a process execution (e.g., the occurrence of a particular activity). These conditions have never been met at the start of any process execution. For each process execution in the training, validation and test set, only the prefixes in which the condition c had not yet been satisfied were considered. The response variables (i.e., the KPIs) in all training, validation and test sets were standardized as discussed in Section 5.2, using the means and standard deviations of the values from the training sets to preserve the integrity of the standardization process.

For each KPI and process, we independently optimized the hyperparameters related to the number of message-passing layers, heads, and hidden dimensions (cf. Table 4). To ensure a robust evaluation and mitigate the effects of random initialization, five different models were trained for each process and KPI.

The performance evaluation was based on the mean absolute error (MAE) between the standardized predictions output by the GNN-based models and the corresponding standardized ground-truth values from the test set. This standardized MAE is beneficial to compare performance across different datasets, and also to account for the variability of the distributions of the response variables. Note that the MAE obtained from standardized data can be converted into the original time scale by multiplying it by the standard deviation of the training set. Since the multiplication factor is constant for a given training set, the improvement percentages are independent of being computed on standardized or on non-standardized MAE.

6.4 Evaluation Results

This section discusses our experimental results, which are summarized in Table 5. Recall that the reported metrics represent the mean performance across five trained models, as mentioned in Section 6.3: this ensures a more robust and generalizable estimate. The performance of our proposed framework, referred to as *Global* in the table, was compared against the state-of-the-art approach based on Learning Sets, referred to as *Local*, and also against a single baseline in which the prediction is equal to the response-variable mean value from the training set. Recall that every KPI value is standardized (cf. Section 5.2).

The Order Management process exhibits linearity in the first two proposed KPIs. Both the *Order* and *Item* display a clear linear pattern, where the improvement in loss using the merged learning set directly correlates with the noise level. In contrast, the *Package* KPI shows the least correlation. This irregularity is expected, as package-related activities in the CPN model do not incorporate the θ factor.

The Logistics process exhibits a more consistent behavior across the three different KPIs. When the noise level α is equal to 0, the MAE improvement is significantly lower compared to when $\alpha = 0.1$ and $\alpha = 0.2$. Additionally, for this process, the results using Merged Learning Sets show a greater improvement with respect to the Learning Sets and tp the baseline. This can be attributed to the higher complexity of the event log, where on average a larger number of objects and events participate in process executions, as summarized in Table 2. As a result, the object-centric graph structure proves to be more effective compared to a simpler event log.

Table 5. Performance results are reported for various KPIs, noise levels, and the baseline across the three processes. Values represent the Mean Absolute Error (MAE) of the standardized KPI values. Terms *Global* and *Local* refer to models trained on the Merged Learning Sets and on the individual Learning Sets, respectively. The *Improvement* column shows the MAE improvement achieved by the global approach wrt. the local, while *Std. Dev.* shows the standard deviation of the respective KPI values in the training set, measured in hours, which was used for standardization.

Process	KPI	Noise Factor α	Global	Local	Improvement	Baseline	Std. Dev.
ORDER MANAGEMENT	Time Until Order is fulfilled	0	0.8518	0.8698	2.08%	0.9024	273
		0.1	0.9906	1.0185	2.74%	1.0395	294
		0.2	0.9048	0.9435	4.11%	0.9736	264
	Time until the Item is delivered	0	0.9606	1.0468	8.24%	1.1926	265
		0.1	1.2140	1.3266	8.49%	1.4413	291
		0.2	0.9581	1.2035	20.40%	1.2144	255
	Time until the Package is delivered	0	0.8798	0.8649	-1.73%	0.8927	77
		0.1	0.6698	0.6523	-2.68%	0.7180	73
		0.2	0.6381	0.6409	0.44%	0.6942	77
LOGISTICS	Time until the second to last event in the p.e.	0	0.7128	0.7536	5.41%	0.8111	143.51
		0.1	0.6820	0.7798	12.55%	0.8491	140.98
		0.2	0.6827	0.7819	12.68%	0.8331	143.81
	Time until the TD is fulfilled	0	0.4739	0.6603	28.22%	0.8087	0.00019
		0.1	0.3054	0.6646	54.05%	0.8330	0.65
		0.2	0.2865	0.6881	58.36%	0.8379	1.31
	Time until Container is ready to depart	0	0.5533	0.7556	26.78%	0.8200	182.38
		0.1	0.3872	0.7400	47.68%	0.8348	179.96
		0.2	0.3820	0.7692	50.35%	0.8378	180.83
IoT	Time until the Pickup Plan is fulfilled	0	0.5777	0.6089	5.13%	0.7206	60
		0.1	0.6163	0.6623	6.95%	0.7264	67
		0.2	0.8008	0.8397	4.64%	0.9147	77

The IoT process instead shows an average level of improvement of approximately 4-6%, which is uncorrelated with the noise factor. This is likely related to the nature of the process. However, when comparing the best-performing models for both Learning Sets and Merged Learning Sets, we observe that the trend correlating improvement with noise remains consistent. The improvements recorded are 4.30%, 4.48% and 7.19% for noise factors of 0, 0.1 and 0.2, respectively.

To summarize, we observed that our framework based on merged learning sets performed better than techniques that are based on non-merged learning sets. In contexts where there is a correlation between parallel process executions, our proposed solution consistently outperforms the traditional approaches in which the learning sets are not merged. The results in Table 5 point out that the accuracy improvement of our framework based on merged learning set is progressively larger as the amount of noise increases. Sometimes, the MAE improvement can be up to and even more than 50%, which notably occurs when process executions are larger in size and chacterized by a higher noise factor. Consider, for example, two KPIs of the logistics datasets with noise

factor α equal to 0.1 and 0.2: the average size of the process executions is twice or even more than the size of the other two processes (cf. Table 2).

In conclusion, this makes our proposed solution more robust to real-world scenarios, where global correlations may exist but are not explicit, and where latent factors, mimicked by noise, can hinder performance if not appropriately handled.

6.5 Time-Complexity Analysis

Graph Attention Networks use attention mechanisms, where a weight is associated with each pair of neighboring nodes [15]. Therefore, the time complexity to train on a graph (N, A) is $O(|N|^2)$ in the worst-case scenario (i.e., a fully-connected graph). In reality, nodes have a limited number of neighbors: the complexity is thus $O(|N| \cdot d)$ where d is the average degree of the graph's nodes. The complexity scales linearly with the number M of graphs, leading to a total time complexity of $O(M \cdot |N| \cdot d)$.

Let us consider the specific setting of predictive monitoring of object-centric processes. Given a process execution p and a timestamp t, let us denote the number of nodes of a prefix $Prefix(p, t)$ with $\overline{M}(p, t)$. Let us indicate the average size of the graph related to a prefix of a process execution with $\overline{P} = avg_{t \in \mathcal{T}}\ avg_{p \in P_{learning}}\ \overline{M}(p, t)$.

Definition 7 describes the size and number of graphs for the non-merged learning sets: the number of graphs is $M = |\mathcal{T}| \cdot |P_{learning}|$ with a certain average size $\overline{P}$. Therefore, for the set of non-merged learning sets, the time complexity is $O(|\mathcal{T}| \cdot |P_{learning}| \cdot \overline{P} \cdot d_n)$ for some average degree d_n.

Definition 8 illustrates the size and number of graphs for the merged learning sets (namely the global-prediction setting): the number of graphs is $M = |\mathcal{T}|$ while the average size is $|P_{learning}| \cdot \overline{P}$ in the worst-case scenario because the merged graph contains $|P_{learning}|$ sub-graphs, each with an average of $\overline{P}$ nodes. Therefore, for the merged learning sets, the time complexity is $O(|\mathcal{T}| \cdot |P_{learning}| \cdot \overline{P} \cdot d_g)$ for some average node degree d_g.

Furthermore, $d_g \simeq d_n$ because, for each graph of the merged learning sets, the sub-graphs tend to form disconnected components or, at most, share a couple of nodes, which does not significantly alter the average degree. It follows that *the asymptotic time complexity of training on learning sets coincides with the case of training on merged learning sets*. In fact, the practical experiments confirm that the difference in training time is negligible in the two settings.

7 Discussion of Potential Limitations and Future Work

This section discusses potential limitations and consequent avenues for future work. Our framework operationalization is based on Graph Neural Networks where categorical features (e.g., the activity names) have been represented via one-hot encoding. Although one-hot encoded vectors are mostly zeros and, thus, can be efficiently stored and processed using sparse matrix representations, this encoding might potentially not scale when there are very large numbers of values of categorical features (e.g., hundreds of distinct activities and objects).

Although this issue did not arise in our experiments, we intend to explore the use of more compact embeddings, such as those proposed by Rama-Maneiro et al. [12], to evaluate potential improvements in scalability and accuracy. However, Rama-Maneiro et al. [12] leverage a numerical scale to produce compact embeddings, which - if not carefully managed - may introduce an implicit ordering that can cause miscalculations of the attention coefficients of our GATs. On the contrary, a one-hot encoding would inherently prevent the risk because no ordering is imposed.

Our embedding based on one-hot encoding does not support categories in the test set that were not observed at training time. This might be a potential limitation when predictive monitoring is used in production, if the process undergoes a concept drift. We plan to investigate proposals on how one-hot encoding can be adjusted to tackle this problem, such as proposed by Roider et al. [13].

The evaluation has only focused on time-related KPIs and performed on synthetic object-centric event logs. However, the underlying theoretical framework is general and can support other KPI categories, which we plan to explore in future work. We also aim to extend the evaluation to real-world object-centric event data, although real data does not allow for verifying the presence of interdependencies and assessing their impact on prediction accuracy.

8 Conclusion

Predictive process monitoring aims to forecast the future state of ongoing process executions using historical event data. Traditionally, it has been applied under the assumption that process executions follow a sequential flow, where each execution is independent and associated with non-shared data objects. However, real-world processes are often more complex, involving multiple objects of different types that interact across different executions. This has led to the introduction of object-centric process monitoring, which acknowledges these interdependencies and provides a more accurate representation of process behavior.

In response to this paradigm shift, recent research has introduced techniques that replace sequence-based models with graph-based representations to capture the relationships between objects. However, existing object-centric predictive process monitoring approaches still overlook certain global influences that are not explicitly modeled through object interactions. For example, in a supply chain scenario, factors like bottlenecks, shared transportation resources, or fluctuating shipment volumes impact process outcomes but may not be captured through object interaction. This limitation can lead to less accurate predictions, as important external dependencies remain unaccounted for.

To address these challenges, this work proposes a new structural framework that merges multiple process execution graphs into a single global representation. By leveraging Graph Neural Networks, the framework enables a unified approach to predictive modeling. Experimental results demonstrate that these global prediction models enhance predictive accuracy, compared to state-of-the-art techniques that do not account for the hidden interferences among process executions, which are not explicitly represented via object interactions.

Acknowledgement. Authors would like to thank Dr. Alessandro Padella, postdoctoral researcher at University of Padua, for his valuable feedback on the assessment results.

References

1. Aalst, W.M.P.: Object-Centric Process Mining: Dealing with Divergence and Convergence in Event Data, LNCS, vol. 11724, pp. 3–25. Springer (2019)
2. Adams, J.N., Drescher, H., Swoboda, A., Günnemann, N., Park, G., van der Aalst, W.: Improving predictive process monitoring using object-centric process mining. In: ECIS 2024 Proceedings. No. 7 (2024)
3. Adams, J.N., Park, G., Levich, S., Schuster, D., van der Aalst, W.M.P.: A framework for extracting and encoding features from object-centric event data. In: Service-Oriented Computing. pp. 36–53. Springer Nature Switzerland (2022)
4. Adams, J.N., Park, G., van der Aalst, W.M.: Preserving complex object-centric graph structures to improve machine learning tasks in process mining. Engineering Applications of Artificial Intelligence **125**, 106764 (2023)
5. Adams, J.N., Schuster, D., Schmitz, S., Schuh, G., van der Aalst, W.M.: Defining cases and variants for object-centric event data. In: 2022 4th International Conference on Process Mining (ICPM). pp. 128–135 (2022)
6. Di Francescomarino, C., Ghidini, C.: Predictive Process Monitoring, pp. 320–346. Springer International Publishing (2022)
7. Fahland, D.: Process Mining over Multiple Behavioral Dimensions with Event Knowledge Graphs, pp. 274–319. Springer International Publishing (2022)
8. Galanti, R., de Leoni, M., Navarin, N., Marazzi, A.: Object-centric process predictive analytics. Expert Systems with Applications **213**, 119173 (2023)
9. Gherissi, W., El Haddad, J., Grigori, D.: Object-centric predictive process monitoring. In: Service-Oriented Computing – ICSOC 2022 Workshops. Springer (2023)
10. Kuhn, M., Johnson, K.: Applied Predictive Modeling. Springer, New York, NY (2013)
11. Object-Centric Event Log Standard: Ocel: Object-centric event log standard. https://www.ocel-standard.org (2024)
12. Rama-Maneiro, E., Vidal, J.C., Lama, M., Monteagudo-Lago, P.: Exploiting recurrent graph neural networks for suffix prediction in predictive monitoring. Computing **106**, 3085–3111 (2024). https://doi.org/10.1007/s00607-024-01315-9
13. Roider, J., Wang, W., Zanca, D., Matzner, M., Eskofier, B.M.: Predictions in predictive process monitoring with previously unseen categorical values. In: Process Mining Workshops (ICPM 2024), Lecture Notes in Business Information Processing, vol. 533, pp. 227–239. Springer (2025). https://doi.org/10.1007/978-3-031-82225-4_17
14. Smit, T.K., Reijers, H.A., Lu, X.: Hoeg: A new approach for object-centric predictive process monitoring. In: Advanced Information Systems Engineering. Springer (2024)
15. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: International Conference on Learning Representations (2018)
16. Wei, J., Ouyang, C., Ma, W., Jiang, D., Xia, J., ter Hofstede, A., Wang, Y., Huang, L.: From Conventional to IoT-Enhanced: Simulated Object-Centric Event Logs for Real-Life Logistics Processes. In: Proceedings of the Best Dissertation Award, Doctoral Consortium, and Demonstration and Resources Forum at BPM 2024, pp. 106–110 (2024)