Connecting the Dots: Path-Based Entity Linking in Object-Centric Event Data

Aaron Küsters and Wil M.P. van der Aalst

Chair of Process and Data Science (PADS), RWTH Aachen University
{kuesters,wvdaalst}@pads.rwth-aachen.de

Abstract. How long does it take from placing an order to its full delivery, and does the number of packages it was shipped in affect this duration? In object-centric event data, answering such questions requires connecting objects and events that have no direct relationship but are linked through chains of intermediate entities. Existing techniques analyze control flow within individual object types yet provide no systematic means to discover or quantify these cross-type connections. We introduce *path schemas*: type-level descriptions of how any two entities (events or objects) can be connected through the data graph. We automatically enumerate path schemas from a compact type graph and rank them by metrics such as selectivity and coverage to separate meaningful connections from noise. Path schemas enable end-to-end performance analysis across object types (e.g., order-to-delivery throughput) and structural connection discovery (e.g., tracing how orders relate to packages through items). We evaluate our open-source implementation on five datasets, including two real-life ones, and show that path schemas uncover process structure often overlooked by existing techniques and that targeted analysis scales to logs with over a million events.

Keywords: Object-Centric Process Mining · Event Knowledge Graph · Path Schemas · Entity Linking · Performance Analysis

1 Introduction

Real-world business processes often involve multiple, interacting objects. For instance, an order might comprise several items, where each item is first picked and packed into a package, which is then shipped to the customer. While for decades, techniques have focused on analyzing processes from a single fixed perspective (e.g., looking at each order in isolation), more recently approaches analyzing multiple perspectives have gained popularity [1,6]. Many of these techniques are based on the notion of *Object-Centric Event Data* (OCED) [1], where each event can relate to multiple objects of different types, and objects can relate to each other, forming a rich graph of entities and relationships. With that flexibility, a natural question arises: *How can we systematically connect entities (events or objects) that are not directly related but are linked through chains of intermediate hops?* For example, measuring order-to-delivery throughput requires connecting

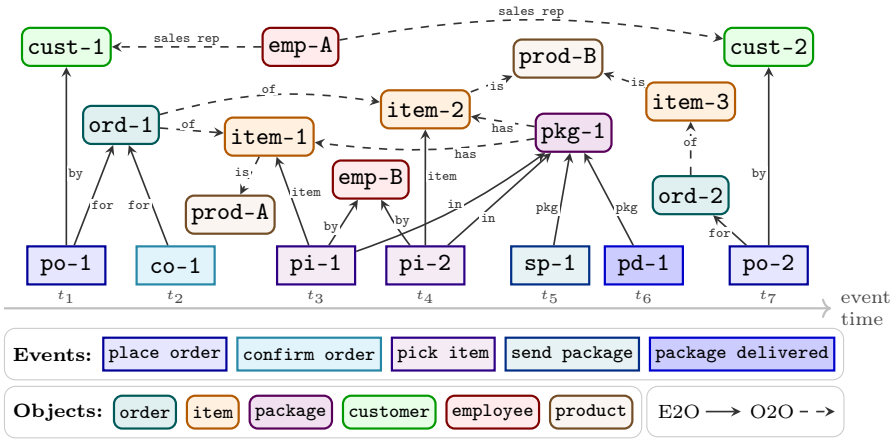


Fig. 1: Instance-level graph of object-centric event data, showing an order management process. Multiple (undirected) paths connect `po-1` and `pd-1`, e.g., over `item-1`, `pi-2`, or even `cust-2`.

a `place order` event to a `package delivered` event through intermediate steps, e.g., `item` and `package` objects. More generally, the analytical task is to systematically identify, quantify, and compare these indirect connections between a chosen source and target entity type, instead of manually guessing which intermediate objects link the two. Techniques like Object-Centric Petri Nets [1] or OC-DFGs [18] often focus on control flow *within* an object type and do not discover or quantify such indirect, cross-type connections. Moreover, they require discovering a complete process model before providing meaningful insights, which can be challenging for complex real-life processes.

Motivating Example. Consider Figure 1: Event `place order` (`po-1`) relates to an `order`, which relates to two `items`; event `package delivered` (`pd-1`) relates to a `package`. Although `po-1` and `pd-1` share no object, they are connected through the order's items and the delivery's package. We call a type-level description of such a connection a *path schema*: a sequence of entity types linked by qualified, directed steps ($>$ forward, $<$ reverse). Schema P connects each `place order` event to related `package delivered` events via three intermediate types:

Multiple schemas can connect the same entity types, differing vastly in quality: some yield tight, meaningful connections while others produce noise. In Section 3, we show that path schemas automatically identify order fragmentation across packages as strongly associated with delivery delay, entirely unsupervised.

Path schemas are versatile analytical primitives that enable various applications for object-centric event data:

– **Cross-Type Performance Analysis:** Connecting events across types enables end-to-end throughput measurement (e.g., order placement to delivery), where different schemas reveal distinct performance characteristics.

- **Structural Connection Discovery:** Connection patterns reveal the relational structure of a process (e.g., how orders map to packages); equivalence classes can correspond to structural invariants.
- **Correlation Analysis:** Connection patterns can be correlated with performance measures to identify structural factors associated with delays or undesired process behavior.

Path schemas can describe links between any pair of entity types (E2E, O2O, E2O, O2E). We primarily evaluate E2E connections to demonstrate their potential for performance analysis.

Analysis Workflow. Given a source and target entity type of interest, the analyst (1) enumerates candidate path schemas from the type graph, (2) ranks and prunes them using structural metrics such as selectivity and coverage, (3) applies temporal and event-selection filters to fix causal direction and performance semantics, (4) inspects the top-ranked schemas, and (5) uses them for end-to-end performance measurement or unsupervised correlation analysis.

Contributions. Our contributions are as follows:

1. We formalize *Instance* and *Type Graphs* and introduce *Path Schemas* as type-level descriptions of connection patterns between any pair of entity types.
2. We define properties (dead schemas, equivalence), metrics (selectivity, coverage, reach, exclusivity), and filters (temporal constraints, event selection) to assess and refine schemas.
3. We provide an open-source implementation of path schemas in Rust4PM, and integrate it into the OCPQ tool. Moreover, we evaluate our approach on five datasets, showing that path schemas surface and rank cross-type process structure that existing techniques cannot discover automatically, and that analysis scales to logs with millions of events.

The remainder is organized as follows. Section 2 introduces path schemas and their metrics, Section 3 presents the evaluation, Section 4 the related work, and Section 5 concludes.

2 Type and Instance Paths

Let $\mathcal{ET}$, $\mathcal{OT}$, $\mathcal{Q}$ be universes of event types, object types, and qualifiers; $\mathcal{E}$, $\mathcal{O}$ universes of events and objects; $\mathbb{T}$ timestamps; and $\mathcal{P}(X)$ the power set of X.

Definition 1 (OCED). Object-Centric Event Data *(OCED) is a tuple* $L = (T_E, T_O, Q, E, O, \mathit{type}, \mathit{time}, R)$*, where* $T_E \subseteq \mathcal{ET}$ *and* $T_O \subseteq \mathcal{OT}$ *are event and object types,* $E \subseteq \mathcal{E}$ *is a set of events,* $O \subseteq \mathcal{O}$ *a set of objects,* $Q \subseteq \mathcal{Q}$ *is a set of qualifiers (relationship types) used,* $\mathit{type}\colon E \cup O \to T_E \cup T_O$ *maps events and objects to their type, such that* $\mathit{type}(e) \in T_E$ *for* $e \in E$ *and* $\mathit{type}(o) \in T_O$ *for* $o \in O$*,* $\mathit{time}\colon E \to \mathbb{T}$ *assigns timestamps to events, and* $R \subseteq (E \times Q \times O) \cup (O \times Q \times O)$ *specifies qualified event-to-object (E2O) and object-to-object (O2O) relationships.*

2.1 OCED Instance and Type Graph

Definition 2 (OCED Instance Graph). *Given an OCED* $L = (T_E, T_O, Q, E, O, type, time, R)$*, its* Instance Graph *is a directed labeled graph* $G = (V, R, \lambda, t)$*:*
- $V = E \cup O$ *is the set of nodes, representing both events and objects.*
- R *is the set of directed, labeled edges (inherited from* L*).*
- $\lambda \colon V \to T_E \cup T_O$ *maps each node to its type, i.e.,* $\lambda(v) = type(v)$*.*
- $t \colon V \to \mathbb{T} \cup \{\bot\}$*, where* $t(e) = time(e)$ *for* $e \in E$ *and* $t(o) = \bot$ *for* $o \in O$*.*

To reason about connections at a structural level, we lift this to a Type Graph.

Definition 3 (Type Graph). *A* Type Graph *is a directed, labeled graph* $G_T = (V_T, R_T)$ *of object or event types as nodes* $V_T \subseteq \mathcal{ET} \cup \mathcal{OT}$ *and edge relations* $R_T \subseteq (\mathcal{ET} \times \mathcal{Q} \times \mathcal{OT}) \cup (\mathcal{OT} \times \mathcal{Q} \times \mathcal{OT})$ *between event and object types (E2O) or two object types (O2O) labeled with a relationship qualifier.*

The Type Graph of an Instance Graph can be constructed by projecting nodes and edges to their types:

Definition 4 (Type of Instance Graph). *The* Type Graph *of an OCED Instance Graph* $G = (V, R, \lambda, t)$ *is the graph* $G_T = (V_T, R_T)$ *where* $V_T = \{\lambda(v) \mid v \in V\}$ *is the set of type nodes, and* $R_T = \{(\lambda(a), q, \lambda(b)) \mid (a, q, b) \in R\}$ *is the set of edges between types, labeled with qualifiers.*

Figure 2 shows the Type Graph for our running example.

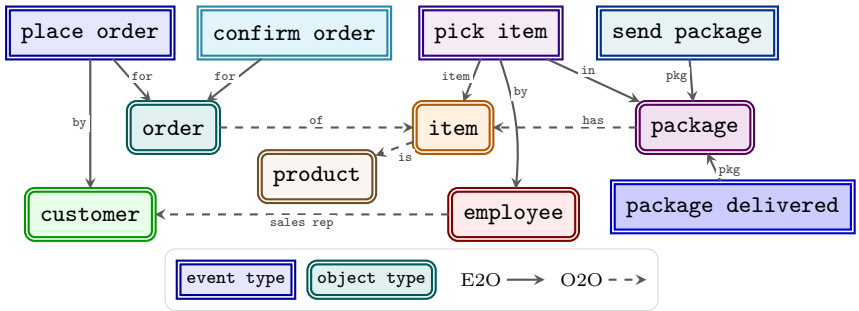


Fig. 2: Type Graph for Figure 1. Double borders denote type-level nodes. The `employee` type exhibits role specialization, causing dead schemas (Definition 8).

2.2 Path Schemas and Their Instances

Intuitively, a path schema is a sequence of entity types where each consecutive pair is connected by a qualifier and a traversal direction (forward $>$ or reverse $<$), as illustrated by schema P in Section 1. The two instance paths of P through concrete entities from Figure 1 are: p_1: ▶ po-1 $>$ ord-1 $>$ item-1 $<$ pkg-1 $<$ pd-1 ■; p_2: ▶ po-1 $>$ ord-1 $>$ item-2 $<$ pkg-1 $<$ pd-1 ■.

Definition 5 (Path Schema). *A* Path Schema *of length* n *over a Type Graph* $G_T = (V_T, R_T)$ *is a sequence* $P = (T_1, s_1, T_2, s_2, \ldots, s_{n-1}, T_n)$*, where:*

- $T_i \in V_T$ *are entity types (nodes in* G_T*) for* $1 \leq i \leq n$.
- *Each step* $s_i = (q_i, d_i)$ *for* $1 \leq i < n$ *consists of a qualifier* $q_i \in \mathcal{Q}$ *and direction* $d_i \in \{>, <\}$ *and must correspond to an edge in* R_T*:*
  - *If* $d_i = >$*, then* $(T_i, q_i, T_{i+1}) \in R_T$.
  - *If* $d_i = <$*, then* $(T_{i+1}, q_i, T_i) \in R_T$.

We write $P_{\blacktriangleright} = T_1$ *and* $P_{\blacksquare} = T_n$ *for the source and target type of* P*, respectively. For* $n = 1$*,* $P = (T_1)$ *is a trivial path schema with no steps.*

We write $k = n-1$ for the number of *steps* (edges) in a path schema and use k throughout the evaluation; a schema with k steps connects $k+1$ entity types. Reverse traversal is crucial, since E2O relationships are directed from events to objects, and connecting back requires following edges in reverse.

Definition 6 (Instance Path). *Let* $G = (V, R, \lambda, t)$ *be an OCED Instance Graph and* $P = (T_1, s_1, T_2, s_2, \ldots, s_{n-1}, T_n)$ *a path schema over its Type Graph. An* Instance Path *p of P is a sequence of entities* $p = (v_1, \ldots, v_n) \in V^n$ *with:*

- ***Type Consistency:*** $\lambda(v_i) = T_i$ *for all* $1 \leq i \leq n$.
- ***Edge Consistency:*** *For each step* $s_i = (q_i, d_i)$ *with* $1 \leq i < n$*, an edge with qualifier* q_i *exists between* v_i *and* v_{i+1} *in the direction specified by* d_i*:*
  - *If* $d_i = >$*, then* $(v_i, q_i, v_{i+1}) \in R$.
  - *If* $d_i = <$*, then* $(v_{i+1}, q_i, v_i) \in R$.
- ***Acyclicity:*** *All entities* $v_1, \ldots, v_n$ *are distinct.*

All instance paths of a path schema can be enumerated for a given dataset. The *connection set* of the resulting paths is the set of endpoints linked through them.

Definition 7 (Instance and Connection Set). *Let* G *be an OCED Instance Graph and* P *a path schema of length* n *on its Type Graph. We write* $\mathcal{I}_P^G$ *for the set of all instance paths of* P *in* G*. Moreover, we define the* connection set *of* P *on* G *as* $C_P^G = \{(v_1, v_n) \mid (v_1, \ldots, v_n) \in \mathcal{I}_P^G\}$*, the set of all endpoint pairs.*

Often, we are interested in instance paths or end entities per source entity. In Figure 1, instance paths p_1 and p_2 for path schema P yield the connection set $\{(\texttt{po-1}, \texttt{pd-1})\}$ and target set $\{\texttt{pd-1}\}$ for source `po-1`.

Attributes. Entity attributes (i.e., object and event attributes) integrate in two complementary ways. As *filters*, attribute predicates can restrict which connections are retained (e.g., only orders above a value threshold), requiring no discretization. As *labels*, a discretized attribute (e.g., region or product category) can replace the entity type as the categorical dimension over which schemas are enumerated and compared, although we focus on the type as the default choice.

2.3 Schema Properties

Not all type-level schemas correspond to actual connections. We identify two structural properties.

Definition 8 (Realized and Dead Schemas). *Let G be an OCED Instance Graph and P a path schema on its Type Graph. We say that P is* realized *if $\mathcal{I}_P^G \neq \emptyset$. Otherwise, i.e., if $\mathcal{I}_P^G = \emptyset$, we say P is* dead. *The* realizability *of a set of schemas $\mathcal{S}$ is $\frac{|\{P \in \mathcal{S} | \mathcal{I}_P^G \neq \emptyset\}|}{|\mathcal{S}|} \in [0, 1]$.*

Dead schemas typically arise from *role specialization*. In Figure 2, the `employee` type connects to both `pick item` (as picker) and `customer` (as sales rep), but these are disjoint populations, so schemas traversing both roles are dead. Low realizability signals that the type graph overapproximates true connectivity.

Definition 9 (Schema Equivalence). *Two schemas P_1, P_2 are* equivalent *($P_1 \equiv P_2$) if their connection set is the same (i.e., $C_{P_1} = C_{P_2}$).*

Equivalence classes help avoid redundant analysis. Moreover, they can correspond to structural constraints: In Figure 1, schemas connecting `place order` to `send package` either via O2O or via event co-occurrence are equivalent, implying that items and their pick events always involve the same package. Violations would indicate a problem (e.g., an item packed in the wrong package).

2.4 Metrics

Realized schemas can still differ vastly: some connect each source to few targets while others produce noise. The *support* $|C_P|$ counts distinct (source, target) pairs; we define four normalized metrics to quantify these differences.

Definition 10 (Schema Metrics). *Given a path schema P, let C_P be the connection set, $\mathcal{S}_P = \{v_1 \mid (v_1, v_n) \in C_P\}$ the active sources, $\mathcal{T}_P = \{v_n \mid (v_1, v_n) \in C_P\}$ the active targets, $N_1 = \{v \in V \mid \lambda(v) = P_{\blacktriangleright}\}$ all source-type entities, and $N_n = \{v \in V \mid \lambda(v) = P_{\blacksquare}\}$ all target-type entities. We define:*

- Coverage*: $cov(P) = |\mathcal{S}_P|/|N_1|$, the fraction of source-type entities that participate as a source in at least one connection.*
- Selectivity*: $sel(P) = |\mathcal{S}_P|/|C_P|$, the ratio of active sources to connections. Equivalently, $1/sel(P)$ is the average number of targets per active source.*
- Reach *(i.e., target-side coverage): $reach(P) = |\mathcal{T}_P|/|N_n|$, the fraction of target-type entities reached by at least one connection.*
- Exclusivity *(i.e., target-side selectivity): $excl(P) = |\mathcal{T}_P|/|C_P|$, the ratio of active targets to connections. Equivalently, $1/excl(P)$ is the average number of sources per active target.*

Figure 3 illustrates these four metrics with a concrete example.

2.5 Temporal Constraints and Event Selection

So far, path schemas treat all edges equally regardless of when events occurred. However, connections where the target event precedes the source rarely represent meaningful process flow. We therefore introduce a filtering mechanism for instance path sets with two concrete families of filters. In the following, let

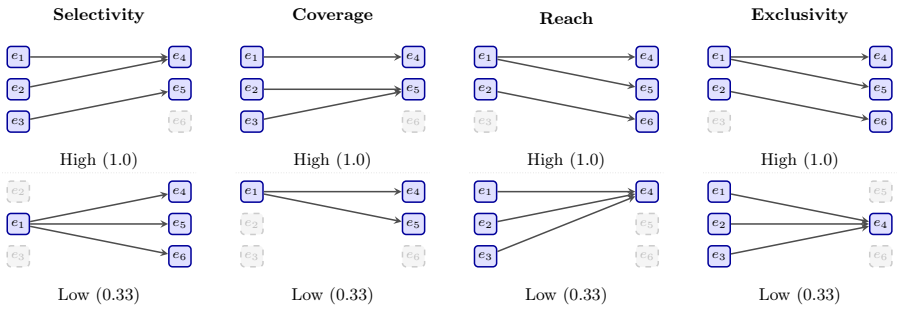


Fig. 3: Illustrations of schema metrics, contrasting low and high values. High *selectivity* connects each source to few targets; low selectivity manifests in a fan-out pattern. High *coverage* means all sources participate; low coverage leaves sources unconnected (dashed). *Reach* mirrors coverage for the target side; high values indicate that most target entities are reached. High *exclusivity* means each target is reached by few sources; low values indicate a fan-in pattern.

$G = (V, R, \lambda, t)$ be a fixed OCED Instance Graph and P a path schema on its Type Graph. As with path schemas, we also write $p_{\blacktriangleright} = v_1$ and $p_{\blacksquare} = v_n$ for an instance path $p = (v_1, \ldots, v_n)$. For a set of instance paths $\mathcal{J}$, we write $\mathcal{J}_v = \{p \in \mathcal{J} \mid p_{\blacktriangleright} = v\}$ for the paths in $\mathcal{J}$ originating from source v.

Definition 11 (Instance Path Filter). *An* instance path filter *on schema P and graph G is a function $f\colon \mathcal{P}(\mathcal{I}_P^G) \to \mathcal{P}(\mathcal{I}_P^G)$ satisfying $f(\mathcal{J}) \subseteq \mathcal{J}$ for all $\mathcal{J} \subseteq \mathcal{I}_P^G$. Filters compose by function composition: $(f_1 \circ f_2)(\mathcal{J}) = f_1\big(f_2(\mathcal{J})\big)$. We write $C_{P,f}^G = \{(p_{\blacktriangleright}, p_{\blacksquare}) \mid p \in f(\mathcal{I}_P^G)\}$ for the* filtered connection set.

Definition 12 (Temporal Filter). *A* temporal filter *restricts instance paths to those satisfying a temporal predicate. For a path $(v_1, \ldots, v_n) \in \mathcal{J}$, let $E_p = \{v_i \mid v_i \in \mathcal{E}\}$ denote its event nodes. We define $\phi_{\to}$ and ϕ_{Δ} (only if $v_1 \in \mathcal{E}$):*
– $\phi_{\to}(\mathcal{J}) = \big\{(v_1, \ldots, v_n) \in \mathcal{J} \;\big|\; \forall\, v_i, v_j \in E_p,\ i < j\colon t(v_i) \le t(v_j)\big\}$ *(*forward*)*
– $\phi_{\Delta}(\mathcal{J}) = \big\{(v_1, \ldots, v_n) \in \mathcal{J} \;\big|\; \forall\, v_i \in E_p\colon |t(v_i) - t(v_1)| \le \Delta\big\}$ *(*bounded*)*

Note that the bounded filter assumes $p_{\blacktriangleright} \in \mathcal{E}$, i.e., that the source is an event with a defined timestamp. The forward filter prunes non-causal connections; the bounded filter limits the time horizon. The second family selects representative targets when a source reaches multiple events.

Definition 13 (Selection Filter). *Let $\mathcal{J}$ be a set of instance paths ending in events, i.e., $p_{\blacksquare} \in \mathcal{E}$ for all $p \in \mathcal{J}$. A* selection filter *selects, for each source, the instance paths whose target satisfies an extremal criterion:*
– $\sigma_{first}(\mathcal{J}) = \big\{p \in \mathcal{J} \;\big|\; t(p_{\blacksquare}) = \min\{t(p'_{\blacksquare}) \mid p' \in \mathcal{J}_{p_{\blacktriangleright}}\}\big\}$ *(earliest)*
– $\sigma_{last}(\mathcal{J}) = \big\{p \in \mathcal{J} \;\big|\; t(p_{\blacksquare}) = \max\{t(p'_{\blacksquare}) \mid p' \in \mathcal{J}_{p_{\blacktriangleright}}\}\big\}$ *(latest)*
If, additionally, $p_{\blacktriangleright} \in \mathcal{E}$ for all $p \in \mathcal{J}$ (E2E), we define: *(soonest)*
– $\sigma_{soonest}(\mathcal{J}) = \big\{p \in \mathcal{J} \;\big|\; |t(p_{\blacksquare}) - t(p_{\blacktriangleright})| = \min\{|t(p'_{\blacksquare}) - t(p_{\blacktriangleright})| \mid p' \in \mathcal{J}_{p_{\blacktriangleright}}\}\big\}$

In practice, a temporal filter ensures causal plausibility, followed by a selection filter. For example, $C_{P,\,\sigma_{last} \circ \phi_{\to}}^G$ yields the last causally related target per

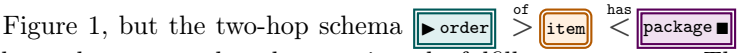
source, useful for measuring when an order is *fully* delivered. For E2E schemas (both source and target are events), the filtered connection set induces a natural performance measure: the *throughput time* of a connection $(e_s, e_t) \in C^G_{P,f}$ is $t(e_t) - t(e_s)$. The schema determines *which* entities are connected; the filter determines *which* connections are retained per source.

Anchored Paths. Selection so far picks an extremal target per source. Instead, an intermediate hop can be used as an *anchor* (e.g., `order` objects), which splits the schema into a source arm and a target arm evaluated independently. Applying σ_{first} or σ_{last} to each arm and pairing the selected events yields one span per anchor entity. This expresses the four first/last combinations, e.g., from the first `place order` to the last `package delivered` of an order, or from the last confirmation to the first shipment. Each arm carries its own metrics.

2.6 Counting Schemas

Beyond connecting pairs of entities, path schemas can also serve as a counting mechanism. For a given source, how many distinct target entities are reachable along a path schema?

Definition 14 (Counting Schema). *The* count *of v_1 under path schema P is $rc_P(v_1) = |\{v_n \mid (v_1, v_n) \in C_P\}|$, the number of target entities reached from v_1.*

For single-hop schemas this reduces to a direct count (e.g., items per order). Multi-hop counting schemas reveal structural information not available from any single relationship. For example, there is no direct `order`→`package` edge in Figure 1, but the two-hop schema 

yields a per-order package count that characterizes the fulfillment structure. The distribution of reachability counts can be correlated with performance measures to identify structural factors associated with delay, as demonstrated in Section 3.

2.7 Algorithms

Schema Enumeration. Given a source type and maximum length k, schemas are enumerated by Breadth-First Search (BFS) on the type graph, treating edges as bidirectional and requiring simple paths, i.e., no repeated types (so a type-revisiting connection, such as an `item` linked to the other `items` in its `package`, is excluded).

Instance Path Finding. For a given schema and source entity, instance paths are found via schema-guided step-wise expansion, extending partial paths one step at a time while pruning acyclicity or temporal violations early. The per-source worst-case complexity is $O(d^k_{\max})$, where $d_{\max}$ is the maximum node degree in the instance graph; in practice, type-guided expansion and selectivity-based pruning keep runtimes well below this bound.

3 Evaluation

We evaluate path schemas along three research questions: **(RQ1)** What structural properties do path schemas exhibit across datasets, and how effectively do metrics separate meaningful schemas from noise? **(RQ2)** How does the approach scale? **(RQ3)** Do path schemas enable cross-type performance analysis and unsupervised discovery of performance correlations?

3.1 Implementation

We implemented path schema discovery in the open-source Rust4PM library [14] with data parallelization. Two optimizations keep discovery tractable: *selectivity-based early termination* maintains an upper bound and stops once it falls below the pruning threshold, and only connection endpoints and timestamps are materialized, not full intermediate paths. We integrate this into the object-centric query tool OCPQ [15] (Figure 4) for interactive analysis: The analyst imports an OCEL 2.0 log [6], selects source and target types on the type graph, and the tool ranks schemas in a sortable table with metric heatmaps. Selecting a schema highlights its path and shows details and throughput; temporal constraints and event-selection strategies can be configured interactively. A schema can also be edited as an OCPQ query, bridging automatic discovery and manual querying. Figure 4 revisits the running example of Section 1 (as one of five RQ3 scenarios, see Table 1). Selectivity ranks the `orders→items` path above the noisy `products` path. The initial implementation, evaluation code and results are publicly available at `https://github.com/aarkue/oced-path-con`.

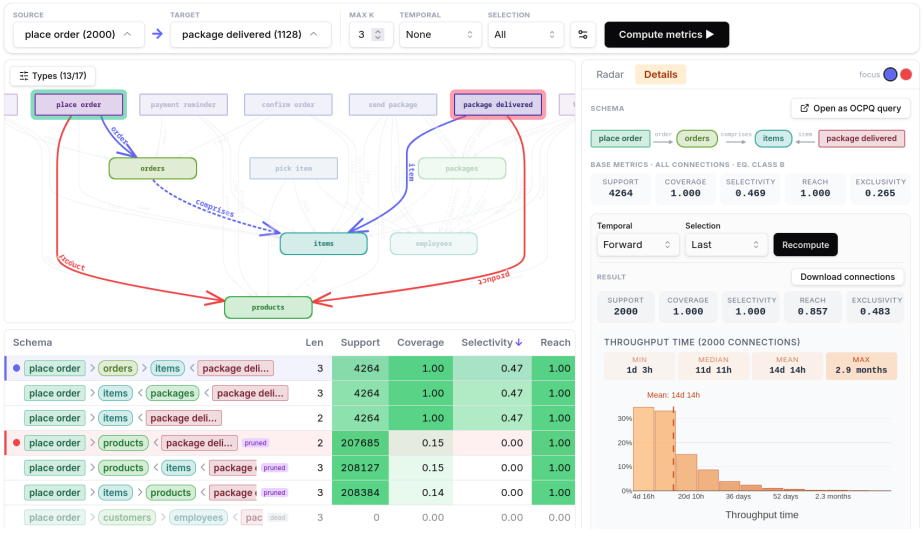


Fig. 4: The `place order` → `package delivered` scenario in the tool. The discriminating `orders→items` schema (blue) and the noisy `products` schema (red) in the type graph. Bottom right: throughput distribution under the *last* filter.

3.2 Datasets and Setup

Beyond this example, we evaluate path schemas systematically: we use five OCEL 2.0 [6] datasets spanning different domains and scales (Table 1). **Order Management (OM)**, **Container Logistics (CL)**, and **Procure-to-Pay (P2P)** are standard simulated object-centric logs covering retail, logistics, and finance, respectively[1]. **BPIC2017** [9] and **BPIC2019** [10] are real-life datasets converted to OCED [13]; at more than 1M events and 100K objects they serve as scalability stress tests. Together, the five datasets form a broad spectrum: Simulated logs provide controlled structural properties (role specialization, hub types, O2O edges), while BPIC2019 lacks O2O relationships entirely, requiring cross-type connections to be recovered through event co-occurrence at k=4. Table 1 also shows one specific E2E scenario (i.e., pair of event types) per dataset that will be further analyzed to address RQ3.

Table 1: An overview of the evaluation datasets and evaluation scenarios.

	Log Characteristics				**Evaluation Scenarios**		
Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{O}\rvert$	$\lvert\mathcal{ET}\rvert$	$\lvert\mathcal{OT}\rvert$	**Scenario**	**Source**	**Target**
P2P	14,671	9,543	10	7	cr → ep	`Create Purch. Req.`	`Execute Payment`
OM	21,008	10,840	11	6	po → pd	`place order`	`package delivered`
CL	35,413	13,910	14	7	ro → dp	`Register Cu. Order`	`Depart`
BPIC2017	1,202,267	106,162	26	4	ca → oa	`A_Create Appl.`	`O_Accepted`
BPIC2019	1,595,923	330,685	41	4	cpr → rgr	`Create Pur.Req.It.`	`Record Goods Rec.`

3.3 Results

RQ1: Structural Properties and Metric Distributions. We enumerated all path schemas across the five datasets for $k \in \{2, 3, 4\}$, pruning schemas with a selectivity below 0.01 early. For BPIC2019 at k=4, exhaustive enumeration of all >2,000 type pairs exceeds practical time limits; we report results based on a 10% stratified sample of type pairs (sampled proportionally across E2E, E2O, O2E, O2O categories). In total, 201,246 schemas were identified, growing from 10,136 at k=2 to 134,356 at k=4 as longer paths traverse more type-graph combinations. Of these, 81,750 (40.6%) are dead, yielding an overall realizability of 0.594. Realizability varies across datasets: P2P (0.988) and CL (0.990) show near-complete realization, OM (0.732) and BPIC2017 (0.906) exhibit moderate rates, while BPIC2019 (0.491) has the lowest, reflecting its large type graph (41 event types) combined with heavy role specialization of the `Resource` object type. A further 37,083 alive schemas (18.4% of all) have selectivity below 0.01 and are pruned as uninformative, leaving 82,413 alive, non-pruned schemas for analysis. Among these, E2E schemas account for 14–83% depending on dataset and k; the remainder are distributed among E2O, O2E, and O2O schemas, confirming that cross-type connections are widespread across all pair kinds.

[1] Sourced from `https://www.ocel-standard.org/event-logs/overview/`.

Dead Schemas serve as a diagnostic: They signal that the type graph over-approximates true connectivity. In OM, a schema from `package delivered` to `place order` via `employees` and `customers` is dead because the `employees` type connects to deliveries as *shipper* and to customers as *sales rep*, but these are disjoint populations. In BPIC2017, `A_Accepted` → `Case_R` → `A_Submitted` is dead while the same pair via `Application` is alive (sel. 1.0). The `Case_R` objects associated with `A_Submitted` events are disjoint from those linked to later lifecycle events. In P2P, a schema linking approval to delegation through shared objects is dead: Requisitions are either approved directly or delegated, but never both. Figure 5 shows metric distributions across the alive, non-pruned schemas. **Selectivity** and **exclusivity** are bimodal, with mass at both extremes, providing the most differentiation between schemas: Schemas traversing a discriminating type (1:1 or 1:few mapping) cluster near 1, while those traversing a resource hub type (1:many) cluster near 0. **Reach** and **coverage** concentrate near 1.0, indicating that most non-pruned schemas cover nearly all source and target entities. BPIC2019 departs from this pattern: Coverage and reach frequently take values near 0, reflecting its structural properties (many infrequent types).

Fig. 5: Distribution of selectivity, reach, exclusivity, and coverage across all alive, non-pruned schemas, each dataset contributing equally.

Equivalence. All datasets exhibit non-trivial equivalence classes. In OM and P2P, more than 75% of type pairs have a compression factor ≥ 2 (i.e., $2r$ path schemas collapse to $\leq r$ equivalence classes). Beyond compression, equivalence classes reveal structural invariants: In OM, three schemas connecting `place order` to `package delivered` (via `items` directly, via `items`→`packages`, and via `orders`→`items`) are equivalent (all sel. 0.47), implying that every item reachable from an order via the O2O relationship also co-occurs with that order in events. If this equivalence did not hold, it would signal items linked to orders through O2O edges but absent from order events, exposing an inconsistency in the data.

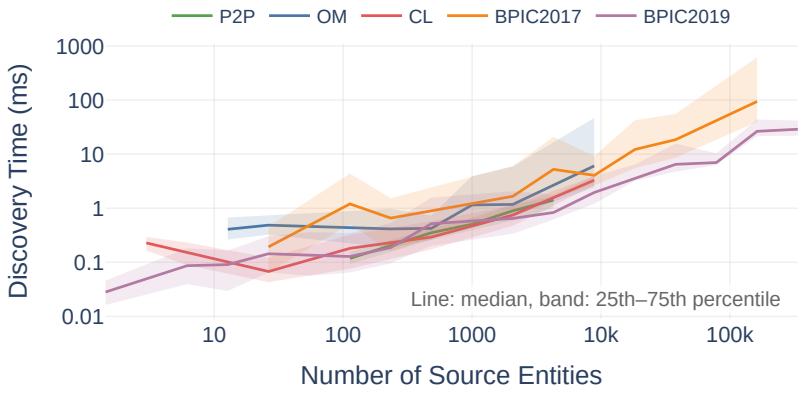


Fig. 6: Discovery time vs. number of source entities (log-log) across all entity type pairs and datasets for $k \in \{2, 3, 4\}$. The median discovery duration and 25th–75th IQR are shown, indicating a linear relationship.

RQ2: Runtime and Scalability. Schema enumeration completes in under 1 ms even at $k = 4$. The cost lies in instance-level connection discovery: Figure 6 plots discovery time against source count for all 82,413 non-pruned schemas, showing a linear trend in log-log space. **Runtimes.** For targeted scenario analysis (including data loading), P2P, OM, and CL complete in under 3 s at k=3; BPIC2017 takes 3.3 s and BPIC2019 takes 60 s. For BPIC2019 at k=4 (466 enumerated schemas), the analysis completes in 5.5 min with a selectivity threshold of 0.05 and 16.3 min with 0.01. Selectivity-based pruning is also effective: At threshold 0.05, 201 of 466 schemas are dead; selectivity pruning cuts the remaining 265 by more than half, leaving only 125 for the analyst to inspect.

RQ3: Cross-Type Performance and Correlation Analysis. We now demonstrate the analysis workflow of Section 1 end to end: metrics rank schemas, temporal and selection filters set causal direction and performance semantics, and counting schemas surface structural correlates of throughput.

Schema Selection and Metrics. Table 2 reports distinct path schemas for the E2E scenarios from Table 1 at $k = 3$, grouped by equivalence classes. Schemas are pruned during discovery when forward-temporal selectivity falls below 0.001, deliberately lower than the 0.01 cutoff of RQ1 so that the table retains low-selectivity schemas for contrast. **Selectivity** is the most discriminating metric. In OM, the schema via `orders`→`items` (sel. 0.47) connects each order to roughly two deliveries, reflecting physical item-to-package routing, while the alternative via `products`→`items` (sel. ≈ 0) links *every* order of the same product to *every* delivery (744K connections). BPIC2017 shows the same contrast: sel. 1.0 via `Application`→`Offer` versus sel. ≈ 0 through shared `Case_R` objects. Similar patterns hold for BPIC2019 (`POItem` sel. 0.92 vs. `Resource` sel. 0.01 with 615K connections). CL has only a single non-dead schema (sel. 0.76), which already provides a meaningful connection. These results highlight where path

Table 2: Path schema metrics (forward temporal, k=3). Schemas with identical metrics are grouped. Fwd./Bnd. red. show path count reduction under forward and bounded (14 d) temporal constraints. Throughput columns show mean throughput time across sources for first/last event selection.

Scenario	Via Types	Supp.	Fwd. red.	Bnd. red.	Metrics (fwd.) Cov.	Rea.	Sel.	Excl.	Time (ms)	Throughput First	Last
cr → ep	Mat., Pur. Ord.	931	0%	92%	1.00	0.80	0.78	1.00	0.3	22.3 d	23.3 d
	Mat., Goods Rec.	537	0%	90%	0.57	0.46	0.77	1.00	0.3	21.6 d	22.6 d
po → pd	Orders, Items	4,264	0%	21%	1.00	1.00	0.47	0.26	0.9	5.3 d	14.6 d
	Products	744,823	48%	92%	1.00	1.00	0.00	0.00	54	0.7 d	228.4 d
ro → dp	Cu. Ord., Tr. Doc.	754	0%	62%	0.95	1.00	0.76	0.17	0.2	16.0 d	19.6 d
ca → oa	Appl., Offer	17,228	0%	54%	0.55	1.00	1.00	1.00	7	18.1 d	18.1 d
	Case_R	155,240	33%	93%	0.02	0.68	0.00	0.08	11	92.5 d	208.8 d
cpr → rgr	POItem	46,752	0%	43%	0.92	0.15	0.92	1.00	15	15.3 d	15.5 d
	Po	748,683	0%	26%	0.95	0.18	0.06	0.08	84	13.6 d	20.7 d
	Resource	615,260	71%	90%	0.11	0.05	0.01	0.02	54	6.4 d	62.6 d

schemas go beyond prior work (cf. Section 4): OPerA [17] requires a shared object between measured events, and OC-DFG analysis [5] stays within one object type. Where a shared object exists (OM via `items`, BPIC2019 via `POItem`), our single-hop schema recovers the same connection these approaches would use. However, the P2P and CL scenarios have no shared object at all and require multi-hop schemas ($k = 3$); the discriminating BPIC2017 schema likewise needs two hops (Application→Offer, $k = 3$), as the only shared object (`Case_R`) yields a near-zero-selectivity connection. Manual EKG queries [11] could express any of these traversals but require the analyst to know the path structure *a priori*; path schemas instead discover and rank all of them automatically, including multi-hop connections that no existing approach surfaces without such prior knowledge. When the relevant path is already known, a manual query may suffice; path schemas help most when many candidate connections must be compared without knowing which is relevant a priori.

Coverage complements selectivity: OM reaches full coverage, while the discriminating BPIC2017 schema covers only 55%, reflecting that not all applications lead to an accepted offer. **Event Selection** captures different performance semantics: In OM, σ_{first} yields 5.3 d mean throughput vs. 14.6 d for σ_{last}. This 9.3-day gap quantifies the additional time until all packages of a multi-item order are delivered. **Temporal Constraints** serve two roles. The forward filter prunes acausal connections: Broad schemas see up to 71% reduction, while discriminating ones see 0%, suggesting that these specific high-selectivity schemas are already temporally coherent. The bounded 14 d filter limits the time horizon and reaches reduction values of 21–93%.

O2O Recovery Through Co-occurrence. BPIC2019 lacks O2O relationships, so cross-type E2E connections require k=4 with event co-occurrence as the bridge. At k=3, schemas traverse a single object type, where `POItem` is the most discriminating (sel. 0.92). At k=4 (threshold 0.01), 466 schemas are enu-

	Counting Schema	ρ	Non-zero %
cr → ep	Cre. PR $\overset{\text{mat}}{\Rightarrow}$ Mat. $\overset{\text{gat}}{\Leftarrow}$ Del. PR Appr.	-0.185	17%
	Cre. PR $\overset{\text{mat}}{\Rightarrow}$ Mat. $\overset{\text{gat}}{\Leftarrow}$ Appr. PR	0.185	83%
	Cre. PR $\overset{\text{mat}}{\Rightarrow}$ Mat.	0.142	100%
po → pd	Pla. Ord. $\overset{\text{it}}{\Rightarrow}$ It. $\overset{\text{con}}{\Leftarrow}$ Pkg.	0.714	100%
	Pla. Ord. $\overset{\text{it}}{\Rightarrow}$ It. $\overset{\text{it}}{\Leftarrow}$ It. OOS	0.709	54%
	Pla. Ord. $\overset{\text{it}}{\Rightarrow}$ It.	0.386	100%
ca → oa	Cr. App. → App. ← App. Incomp.	0.406	73%
	Cr. App. → App. ← App. Valid.	0.290	100%
	Cr. App. → App. → Offer	0.270	100%
cpr → rgr	Cre. PRI → PO ← Clear Inv. → POItem	-0.410	54%
	Cre. PRI → PO ← Clear Inv.	-0.408	54%
	Cre. PRI → PO ← Rec. Inv. Rec. → POItem	-0.406	88%

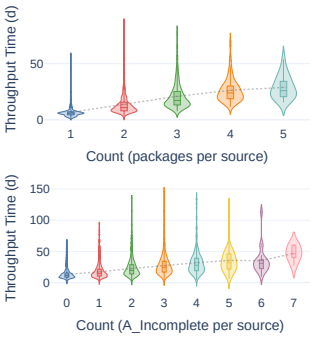


Fig. 7: Automatic correlation discovery. All enumerated counting schemas ranked by Spearman correlation with E2E throughput ($p < 0.01$ for all); equivalent schemas are collapsed. Right: Visualization for the top-ranked OM (top) and BPIC2017 (bottom) counting schema against throughput time.

merated (201 dead, 102 pruned, 163 alive); all high-selectivity schemas connect through `POItem` or `PO`, confirming that meaningful cross-type connections can be recovered from event co-occurrence alone when O2O relationships are absent.

Counting Schema Correlation. For each E2E scenario, we enumerate all counting schemas (Definition 14) from the source event type. Each source event yields one reachability count and one throughput time (σ_{last}); we rank counting schemas by Spearman correlation across source events, collapsing equivalence classes (Figure 7, left). In OM, equivalence classes reduce 37 counting schemas (k=2) to 20 distinct classes. The top-ranked class counts *packages per order* (4 equivalent schemas, $\rho = 0.714$, $p < 0.001$): single-package orders have a median throughput of 5.9 d vs. 26.0 d for 5-package orders (Figure 7, right). No direct `order`–`package` relationship exists; the 2-hop schema through `item` discovers this connection automatically, outperforming the best single-hop baseline (*items per order*, $\rho = 0.386$). The second-strongest predictor is *item out of stock* ($\rho = 0.709$, count $\geq$1 for 54% of sources): orders with at least one out-of-stock item have a mean throughput of 20.3 d vs. 7.9 d otherwise. For BPIC2017, the number of `A_Incomplete` events per application ($\rho = 0.406$, count $\geq$1 for 73% of sources) is the strongest predictor (Figure 7, right), suggesting rework-driven delays where repeated resubmission of incomplete information prolongs processing. For P2P, delegated vs. direct approval schemas show opposing correlations ($\rho = \pm 0.185$), linking the approval route to throughput; for CL, the 1:1 structure yields zero variance, so no correlations emerge. For BPIC2019, the strongest predictor counts `POItems` reachable via `Clear Invoice` on the shared `PO` ($\rho = -0.410$, count $\geq$1 for 54% of sources), likely reflecting that items on larger POs complete faster (11.9 d vs. 19.8 d). All these patterns were discovered automatically without prior domain knowledge of the process structure.

3.4 Threats to Validity

Selectivity and coverage are proxies for connection meaningfulness; no ground truth exists for schema relevance. Still, high-selectivity schemas consistently

yield domain-interpretable connections (e.g., item-to-package routing in OM), while low-selectivity ones produce combinatorial noise. Domain-expert validation remains future work. The RQ3 counting-schema correlations are associative, not causal; we guard against spurious associations with significance thresholds ($p < 0.01$, Bonferroni-corrected), but causal interpretation requires domain knowledge. Three datasets are simulated; the two real-life logs (BPIC2017, BPIC2019) are BPI challenge data converted to OCED rather than originally object-centric. Both still offer realistic scale (> 1M events), and BPIC2019's lack of O2O relationships stress-tests the approach. Validation on object-centric industrial data would strengthen generalizability. We restrict schemas to acyclic paths; bounded revisitation is a natural extension. At $k \geq 5$, combinatorial growth can make exhaustive enumeration impractical.

4 Related Work

Performance analysis is well-established in case-centric process mining, with mature frameworks for defining and computing KPIs over single-case traces [7]. In Object-Centric Process Mining (OCPM) [1], performance analysis is less developed. Most object-centric discovery techniques (OC-DFGs [5], OC Petri nets [1]) model control flow *within* individual object types. OPerA [17] measures sojourn times and synchronization delays within a single object's lifecycle, requiring a perfectly fitting object-centric Petri net as input. Park et al. [18] extend OC-DFGs with waiting and service times at activities. Both approaches compute performance along an object's lifecycle trajectory, so they require that the measured events share a common object; they cannot express durations where no single object participates in both endpoints (e.g., order placement to package delivery, linked only through items). De Leoni and Volpato [16] show that such implicit interferences between object-centric executions significantly affect prediction accuracy, motivating the need to make hidden connections explicit.

More expressive frameworks can capture these cross-type relationships: Esser and Fahland [11] proposed Event Knowledge Graphs (EKGs), storing events and objects as labeled property graphs and supporting hand-crafted Cypher queries for cross-object performance measures. OCPQ [15] provides a visual query language for selecting and querying combinations of objects and events. However, both approaches require manually specifying the exact traversal path, which assumes prior knowledge of the data's structure. A generic Cypher query could enumerate instance paths up to a fixed length without such prior knowledge, but returns raw paths rather than ranked typed schemas, without dead-schema or equivalence detection, temporal filters, or counting schemas. Path schemas automate this by enumerating all type-level schemas from the type graph, which is derived automatically from the log rather than supplied, and ranking them by quality metrics, as demonstrated in Section 3, where metrics rank meaningful routes ahead of spurious ones. Discovered schemas could also serve as a starting point for EKG or OCPQ analysis by generating candidate queries. Table 3 summarizes the positioning of path schemas against these approaches.

Table 3: Positioning against existing OCED approaches for performance analysis. EKG/OCPQ require manual specification of cross-type connections. OC-DFG is based on DFGs built from the data, but requires no further model as input.

	OPerA [17]	OC-DFG [5,18]	EKG [11]/OCPQ [15]	**Path Schemas**
Metric scope	activity	activity pair	flexible	flexible
Cross-type	×	×	✓	✓
Model-free	×	$\sim^{\text{DFG}}$	✓	✓
Auto. discovery	N/A	N/A	×	✓

Prescriptive formalisms such as Proclets [12] and Object-Centric Behavioral Constraints (OCBC) [3] define inter-object interactions at design time. Path schemas take a *descriptive* perspective, discovering how entities connect *across* types and enabling cross-type performance measurement directly from data.

Adams et al. [2] define process executions as connected subgraphs where the analyst selects participating object types. Van Detten et al. [8] generalize this into a framework for case construction. Basmer et al. [4] take a more exploratory approach, materializing views from different context definitions to help analysts discover relevant perspectives on the data. All three define connections at the level of case notions or contexts; path schemas instead operate at the entity-to-entity level, discovering and quantifying specific connection patterns.

The concept of type-level path templates also appears in heterogeneous information networks as meta-paths [19], where selecting meaningful paths for a given task is an active research challenge addressed through learning-based methods. Instead, we apply exhaustive enumeration combined with interpretable quality metrics, providing a ranking without requiring predefined training data. Temporal filters and structural properties like dead schemas and equivalence classes arise specifically from the structure of *object-centric event data*.

5 Conclusion

We introduced path schemas as a new analytical primitive for object-centric process mining that discovers and quantifies cross-type entity connections without requiring a process model or domain knowledge. Our evaluation on five datasets shows that structural properties expose type-graph overapproximation, that the introduced metrics rank discriminating schemas ahead of spurious ones, that reachability counts can reveal structural correlates of performance without supervision, and that targeted analysis scales to logs with millions of events.

Future work. Dead schemas and equivalence classes are promising diagnostics deserving deeper investigation. Further extensions include attribute-awareness, bounded cyclic paths that allow recurring types such as items packaged together, and validation with domain experts on object-centric industrial data.

Acknowledgments. The authors gratefully acknowledge the German Federal Ministry of Research, Technology and Space (BMFTR) and the state government of North Rhine-Westphalia for supporting this work as part of the NHR funding.

References

1. van der Aalst, W.M.P., Berti, A.: Discovering Object-centric Petri Nets. Fundam. Informaticae **175**(1-4), 1–40 (2020)
2. Adams, J.N., Schuster, D., Schmitz, S., Schuh, G., van der Aalst, W.M.P.: Defining cases and variants for object-centric event data. In: ICPM. pp. 128–135. IEEE (2022)
3. Artale, A., Kovtunova, A., Montali, M., van der Aalst, W.M.P.: Modeling and Reasoning over Declarative Data-Aware Processes with Object-Centric Behavioral Constraints. In: BPM. LNCS, vol. 11675, pp. 139–156. Springer (2019)
4. Basmer, M., Ueck, H., Fahland, D., Weidlich, M.: MANTA: materializing views on event data for context exploration in process analysis. In: BPM. LNCS, vol. 16044, pp. 51–68. Springer (2025)
5. Berti, A., van der Aalst, W.M.P.: OC-PM: analyzing object-centric event logs and process models. Int. J. Softw. Tools Technol. Transf. **25**(1), 1–17 (2023)
6. Berti, A., Koren, I., Adams, J.N., Park, G., Knopp, B., Graves, N., Rafiei, M., Liß, L., genannt Unterberg, L.T., Zhang, Y., Schwanen, C.T., Pegoraro, M., van der Aalst, W.M.P.: OCEL (Object-Centric Event Log) 2.0 Specification. CoRR **abs/2403.01975** (2024)
7. del-Río-Ortega, A., Resinas, M., Cabanillas, C., Cortés, A.R.: On the definition and design-time analysis of process performance indicators. Inf. Syst. **38**(4), 470–490 (2013)
8. van Detten, J.N., Schumacher, P., Leemans, S.J.J.: A framework for advanced case notions in object-centric process mining. In: ICPM Workshops. LNBIP, vol. 533, pp. 402–414. Springer (2024)
9. van Dongen, B.: BPI Challenge 2017 (2017), `https://data.4tu.nl/articles/dataset/BPI_Challenge_2017/12696884/1`
10. van Dongen, B.: BPI Challenge 2019 (2019), `https://data.4tu.nl/articles/_/12715853/1`
11. Esser, S., Fahland, D.: Multi-Dimensional Event Data in Graph Databases. Journal on Data Semantics **10**(1-2), 109–141 (2021)
12. Fahland, D.: Describing Behavior of Processes with Many-to-Many Interactions. In: Petri Nets. LNCS, vol. 11522, pp. 3–24. Springer (2019)
13. Khayatbashi, S., Hartig, O., Jalali, A.: Transforming event knowledge graph to object-centric event logs: A comparative study for multi-dimensional process analysis. In: ER. LNCS, vol. 14320, pp. 220–238. Springer (2023)
14. Küsters, A., van der Aalst, W.M.P.: Rust4PM: A versatile process mining library for when performance matters. In: BPM (Demos / Resources Forum). CEUR Workshop Proceedings, vol. 3758, pp. 91–95. CEUR-WS.org (2024)
15. Küsters, A., van der Aalst, W.M.P.: OCPQ: object-centric process querying and constraints. In: RCIS (1). LNBIP, vol. 547, pp. 383–400. Springer (2025)
16. de Leoni, M., Volpato, P.: Global predictive monitoring of object-centric processes. In: BPM. LNCS, vol. 16044, pp. 255–272. Springer (2025)
17. Park, G., Adams, J.N., van der Aalst, W.M.P.: OPerA: Object-centric performance analysis. In: ER. LNCS, vol. 13607, pp. 281–292. Springer (2022)
18. Park, G., Adams, J.N., van der Aalst, W.M.P.: Conformance checking and performance analysis using object-centric directly-follows graphs. In: BPM (Forum). LNBIP, vol. 526, pp. 179–196. Springer (2024)
19. Sun, Y., Han, J., Yan, X., Yu, P.S., Wu, T.: Heterogeneous information networks: the past, the present, and the future. Proc. VLDB Endow. **15**(12), 3807–3811 (2022)