

Connecting the Dots

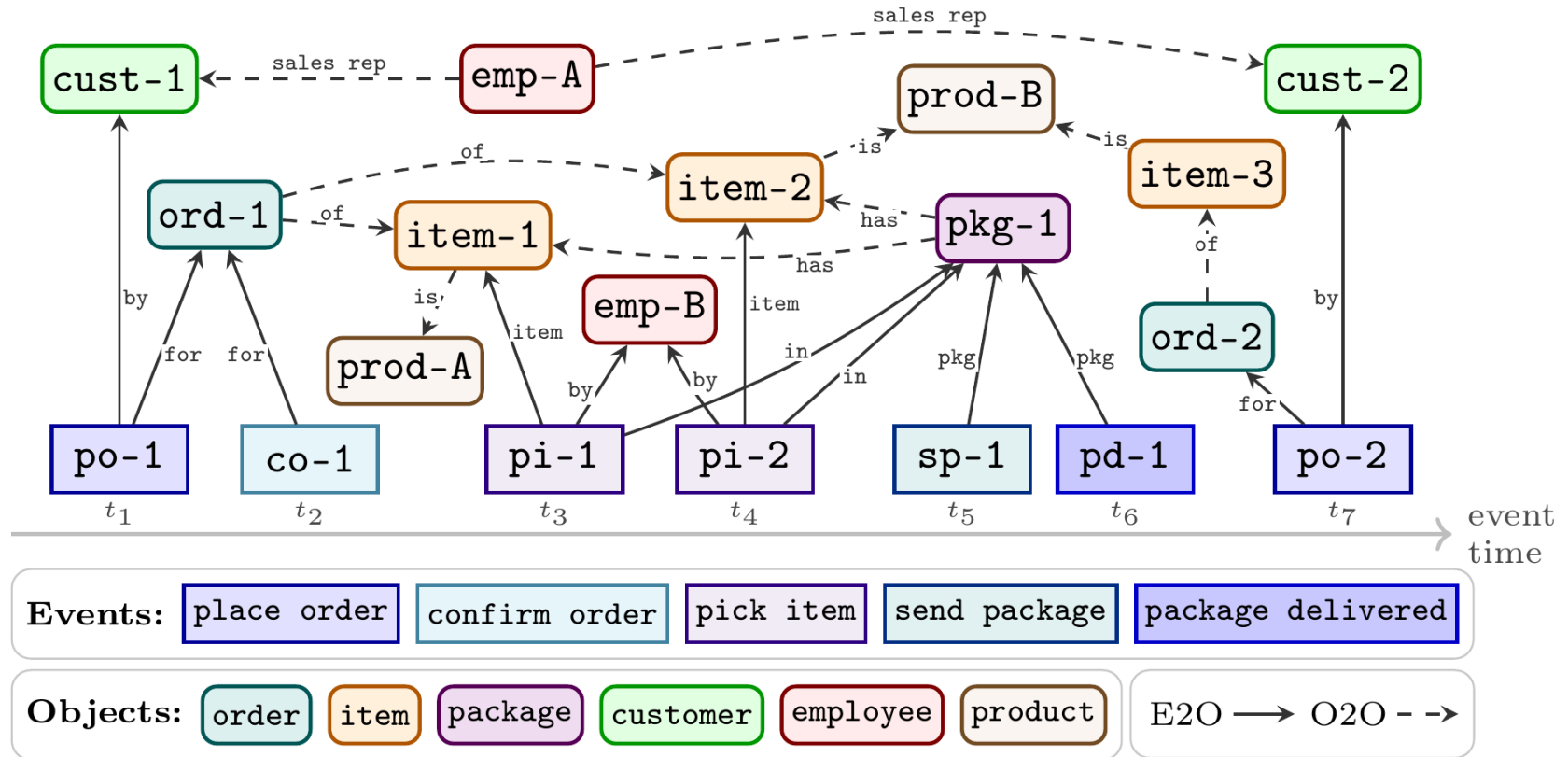
Path-Based Entity Linking in Object-Centric Event Data

Aaron Küsters, Wil M.P. van der Aalst

2026-09-29

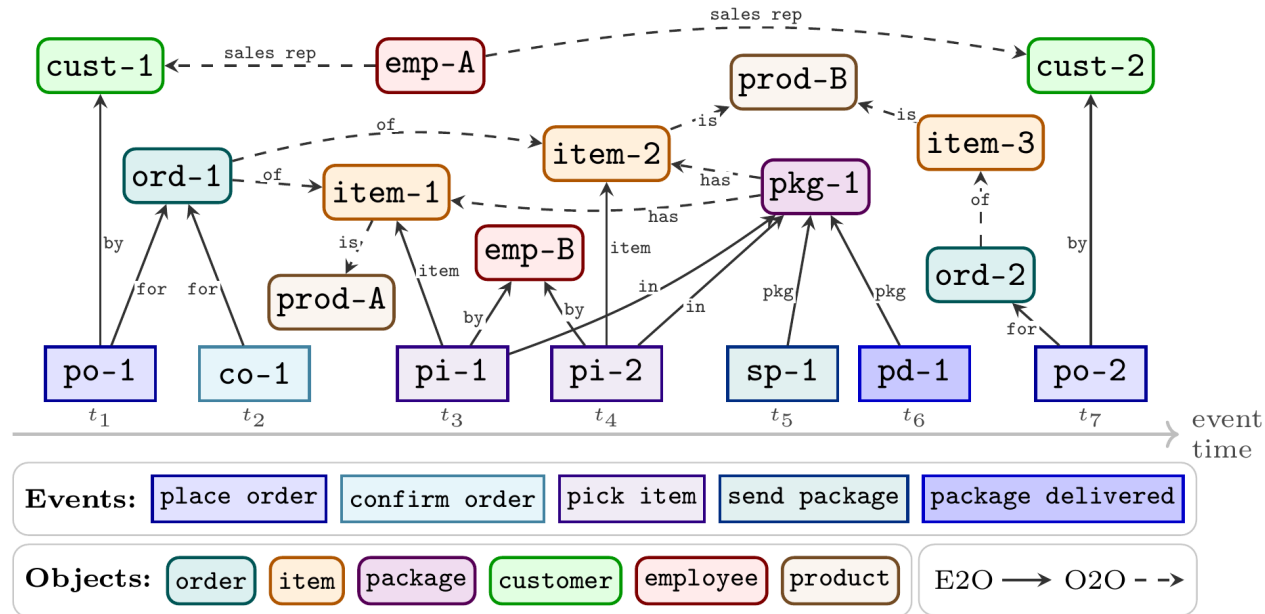
BPM Forum @ BPM 2026

Object-Centric Event Data



Connecting the Dots: Order to delivery

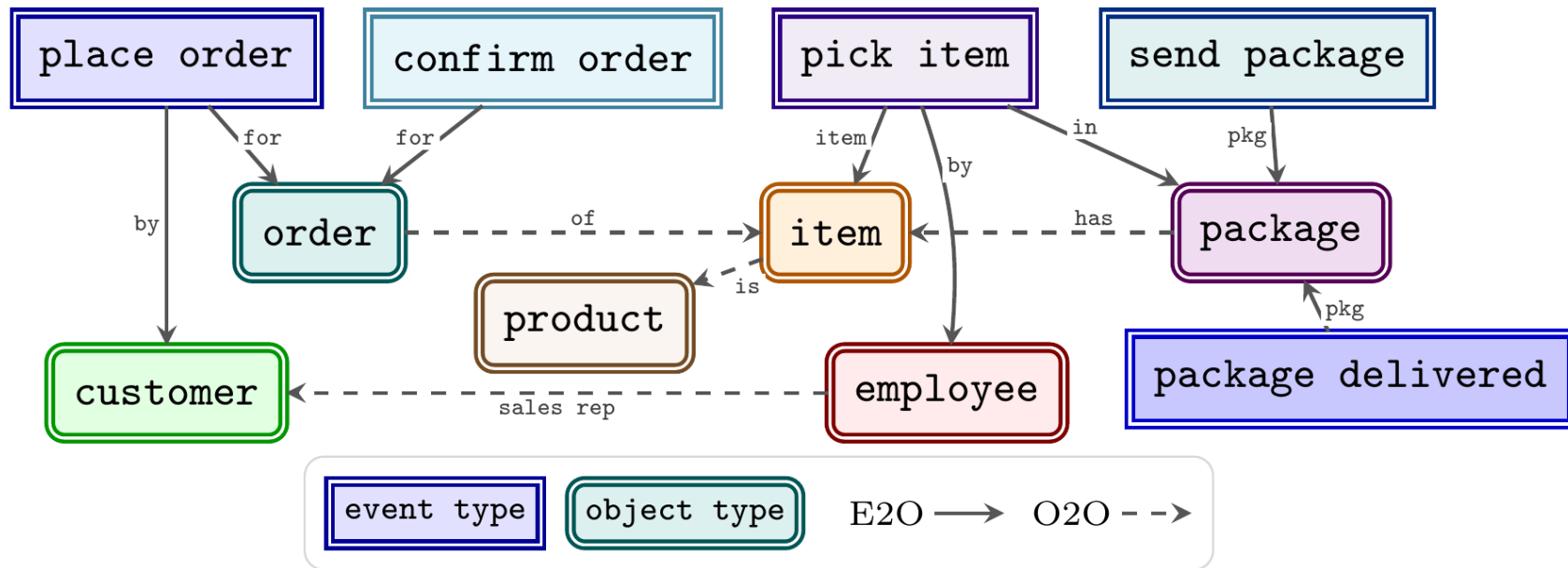
How long from placing an order to its full delivery? Do more packages make it slower?



place_order (po-1) and package_delivered (pd-1) share **no object**.

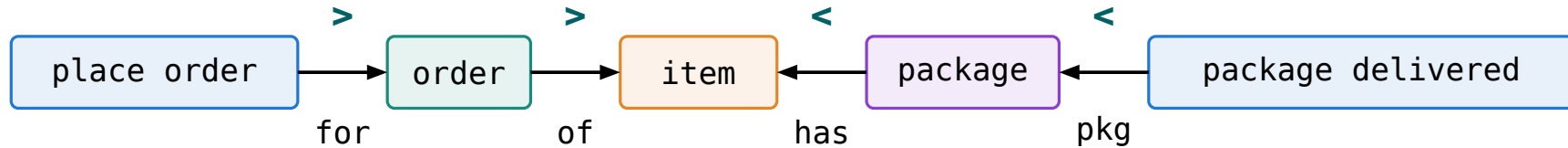
Type graph

Projecting events and objects to their types gives the **type graph**, with edges labeled by qualifier:

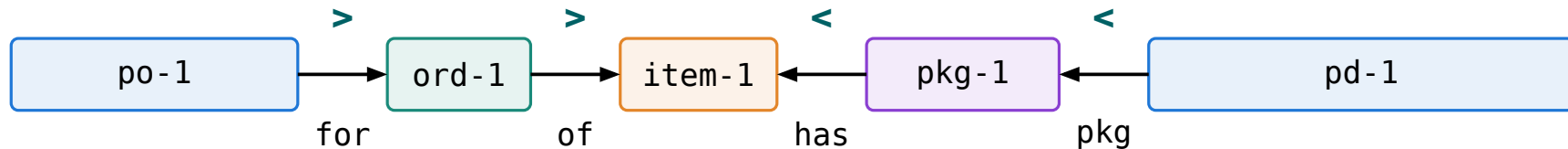


Path schemas

A **path schema** is a path through the type graph. It describes how entities of two types can be connected:



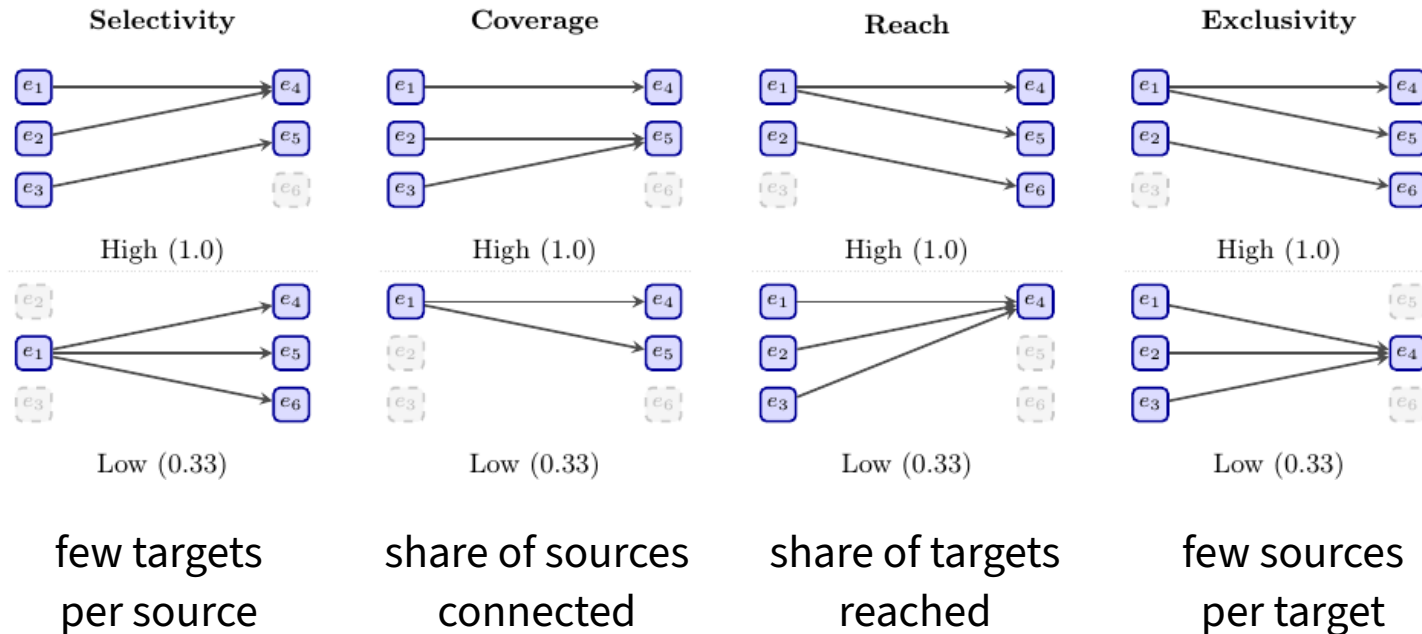
One of its instance paths in the data:



Each step follows a qualified relationship **forward** (>) or **in reverse** (<).

Schema metrics

All schemas up to k steps between two types are enumerated and ranked by metrics:



Ranking schemas

Order management log, place order to package delivered, $k = 3$, two of the schemas:

Schema via	Connections	Coverage	Reach	Selectivity	Exclusivity
order, item	4,264	1.00	1.00	0.47	0.26
product	744,823	1.00	1.00	< 0.01	< 0.01

Both reach every order and every delivery. Selectivity and exclusivity tell them apart:

via order, item: each order reaches about two deliveries.

via product: each order is linked to every later delivery with one of its products, about 370 per order.

Filters and Selectors

Filters decide which connections count:

 **Temporal:** forward in time only, or within a time window

 **Selection:** first or last target event per source

Throughput time

Mean time from place order to package delivered, via order and item:

5.3 d

to the **first** package delivery

14.6 d

to the **last** package delivery

9.3 d

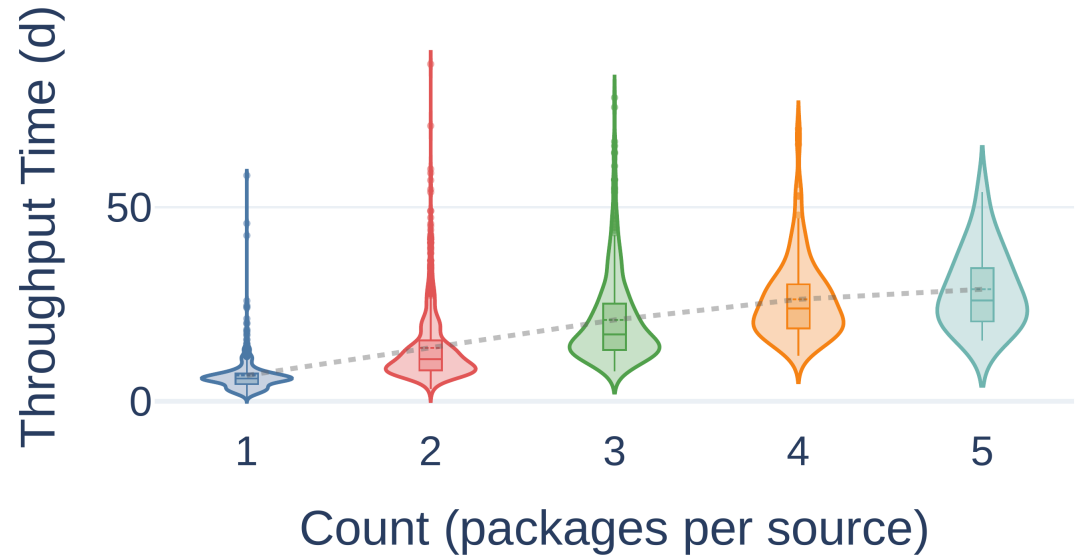
mean gap between first and last delivery

Correlates of delay

Path schemas can also **count**, e.g., packages per order via `item`.

All 37 counting schemas from `place_order`, ranked by Spearman correlation with the time to the last delivery.

Strongest: packages per order, $\rho = 0.714$.
Median 5.9 d for one package, 26.0 d for five.



Evaluation

Five logs: three simulated, BPIC2017 and BPIC2019 with over 1M events each.

201,246

schemas with 2 to 4 steps, all
type pairs

40.6%

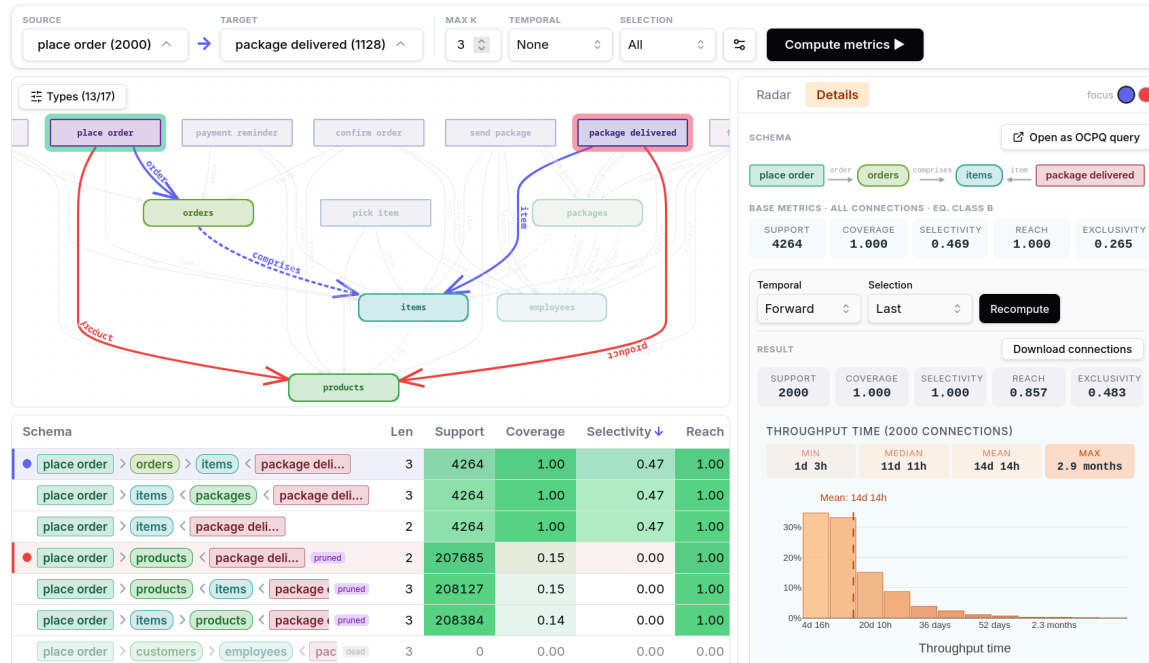
of them never occur in the data

3.3 s

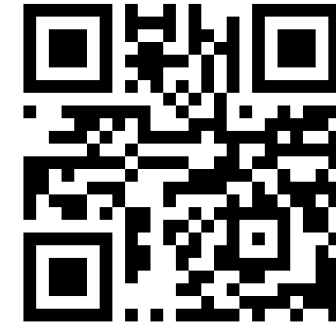
BPIC2017 scenario, $k = 3$,
including loading

-  Discovery time grows at most linearly with the number of source entities.
-  BPIC2019 has no object-to-object relations. With 4 steps, path schemas link its object types through shared events.

OCPQ integration



Pick source and target types, compare the ranked schemas, open one as an OCPQ query.



ocpq.aarkue.eu

Takeaways

-  **Path schemas** link events and objects through chains of intermediate entities, also when they share no object.
-  They are enumerated from the type graph and ranked by metrics, without a process model or a known path.
-  They give end-to-end throughput times across object types and rank correlates of delay, e.g., packages per order.

Thank you

Thanks!

 ocpq.aarkue.eu

 github.com/aarkue/oced-path-con

 kuesters@pads.rwth-aachen.de





Bonus

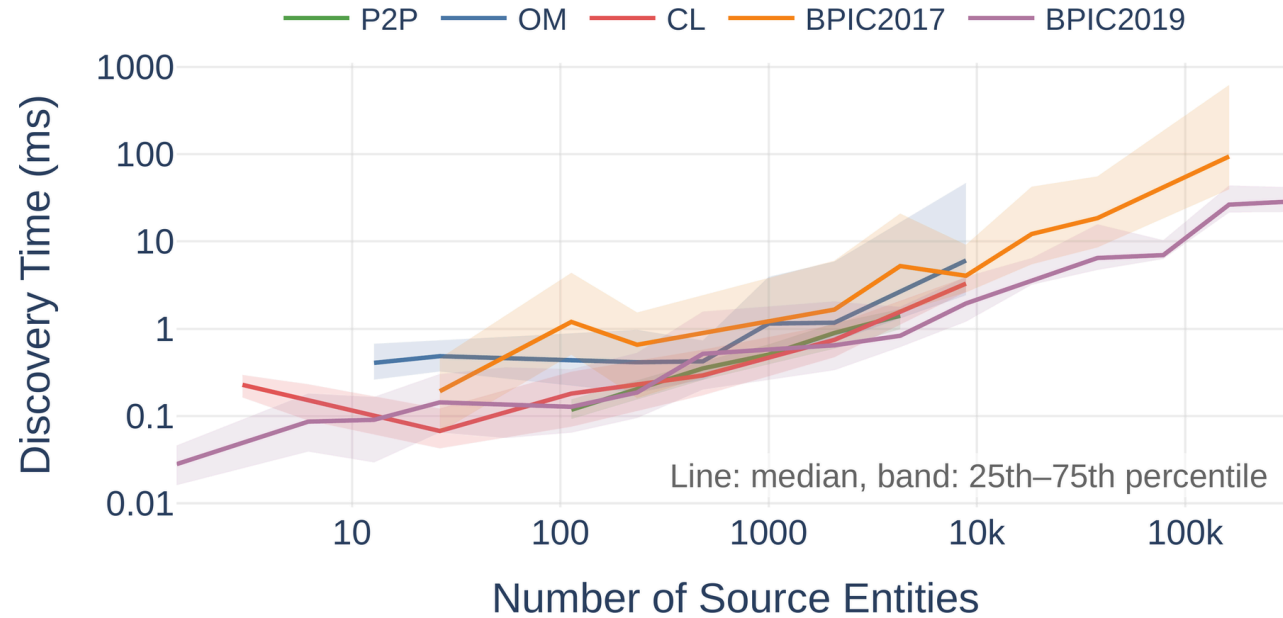
Dead schemas

- ⊘ A schema is **dead** if no instance path exists in the data.
- 👤 Dead schemas typically come from role specialization. employee connects to pick item as picker and to customer as sales rep. These are different employees, so schemas through both roles are dead.
- ℹ Many dead schemas mean the type graph allows more connections than the data has.

Datasets

Log	Events	Objects	Scenario	Source → target
P2P	14,671	9,543	cr → ep	Create Purch. Req. → Execute Payment
OM	21,008	10,840	po → pd	place order → package delivered
CL	35,413	13,910	ro → dp	Register Cu. Order → Depart
BPIC2017	1,202,267	106,162	ca → oa	A_Create Appl. → O_Accepted
BPIC2019	1,595,923	330,685	cpr → rgr	Create Pur.Req.It. → Record Goods Rec.

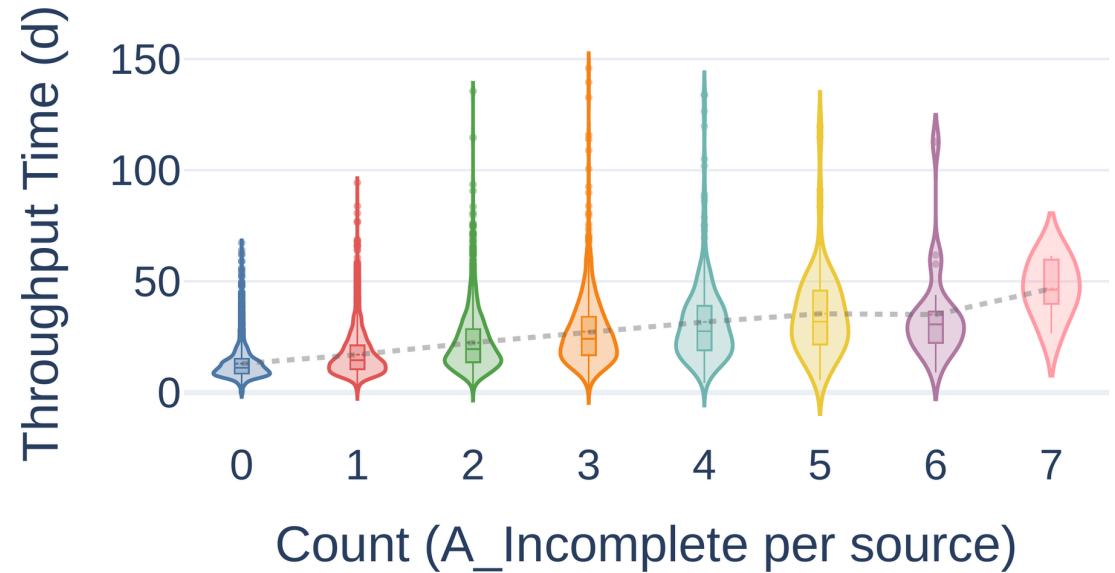
Runtime



Discovery time vs. number of source entities (log-log), all 82,413 non-pruned schemas, $k \in \{2, 3, 4\}$ (BPIC2019 at $k = 4$ from a 10% sample of type pairs).

BPIC2017 correlation

Number of A_Incomplete events per application vs. throughput from A_Create Application to O_Accepted: $\rho = 0.406$. Repeated requests for missing information go along with longer cases.



Existing approaches

-  OC-DFGs and object-centric Petri nets model control flow per object type.
-  OPerA and OC-DFG performance analysis need an object shared by both events. OPerA also needs a perfectly fitting object-centric Petri net.
-  Event knowledge graph and OCPQ queries can follow any path, but the analyst has to know it beforehand.