RDF Graph Alignment with Bisimulation

Sławek Staworko[1,2,3]
(joint work with Peter Buneman[1])

[1]University of Edinburgh, Scotland

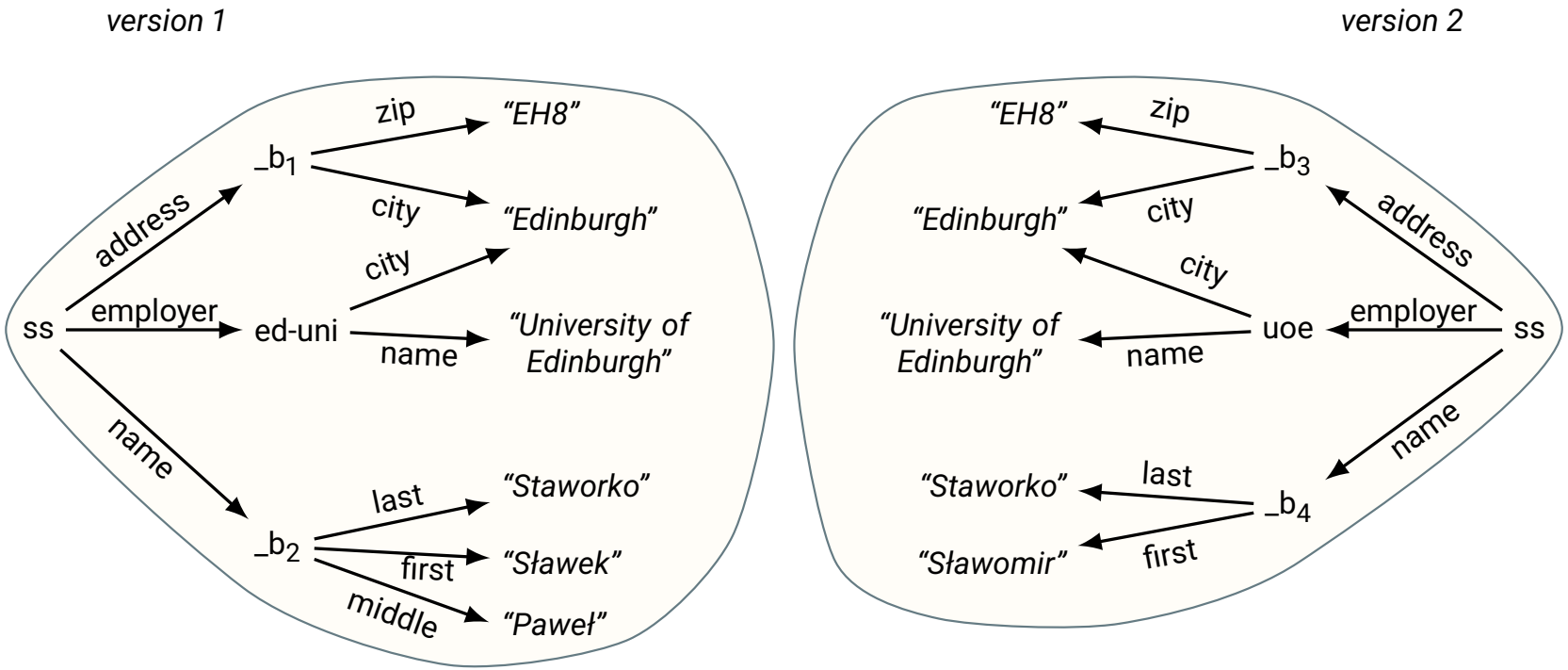
[2]LINKS, INRIA & CNRS, Lille, France

[3]CRIStAL, University of Lille & CNRS, France

VLDB Conference, New Delhi, India

September 8, 2016

Aligning Evolving RDF Graphs

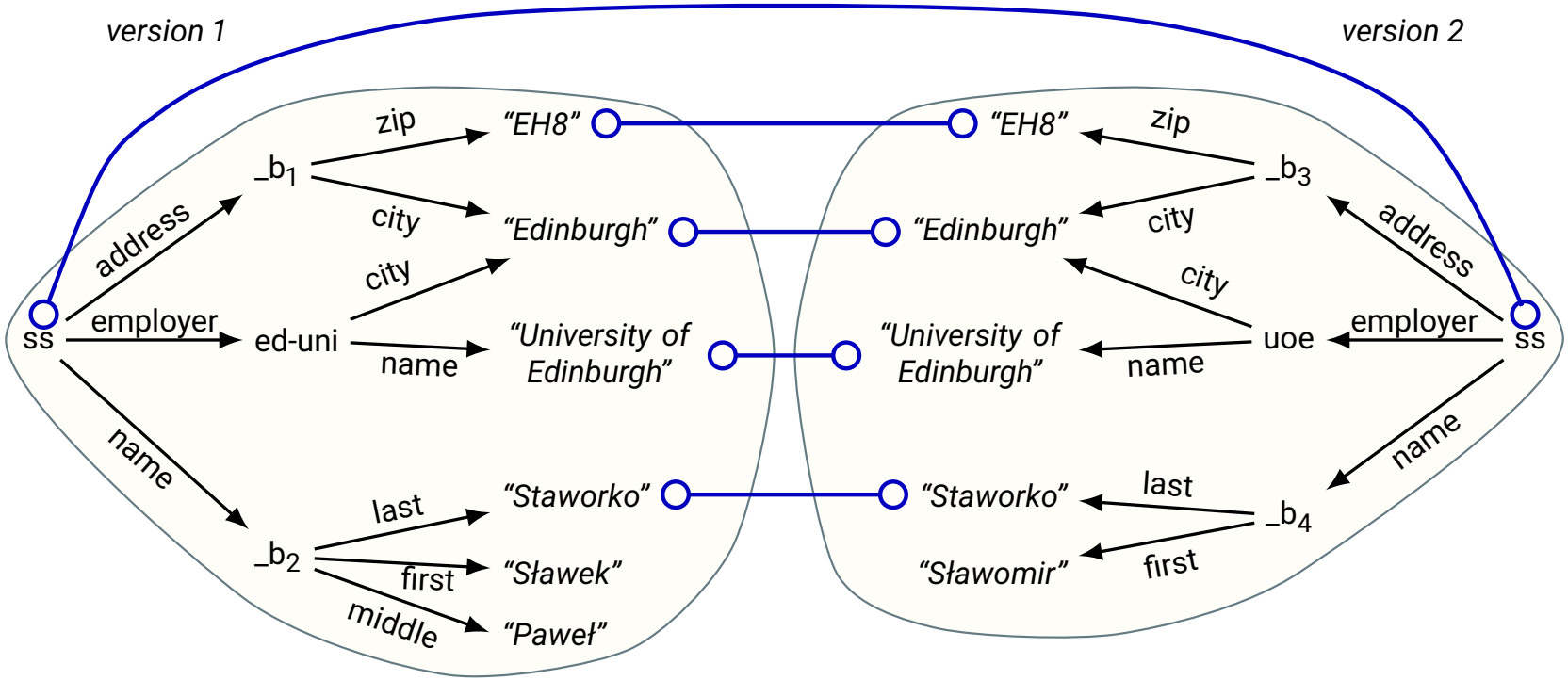


Why is it interesting?

- Version diffs (deltas)
- Efficient storage
- Temporal querying

But what are URIs for?

Aligning Evolving RDF Graphs



Why is it interesting?

- Version diffs (deltas)
- Efficient storage
- Temporal querying

But what are URIs for?

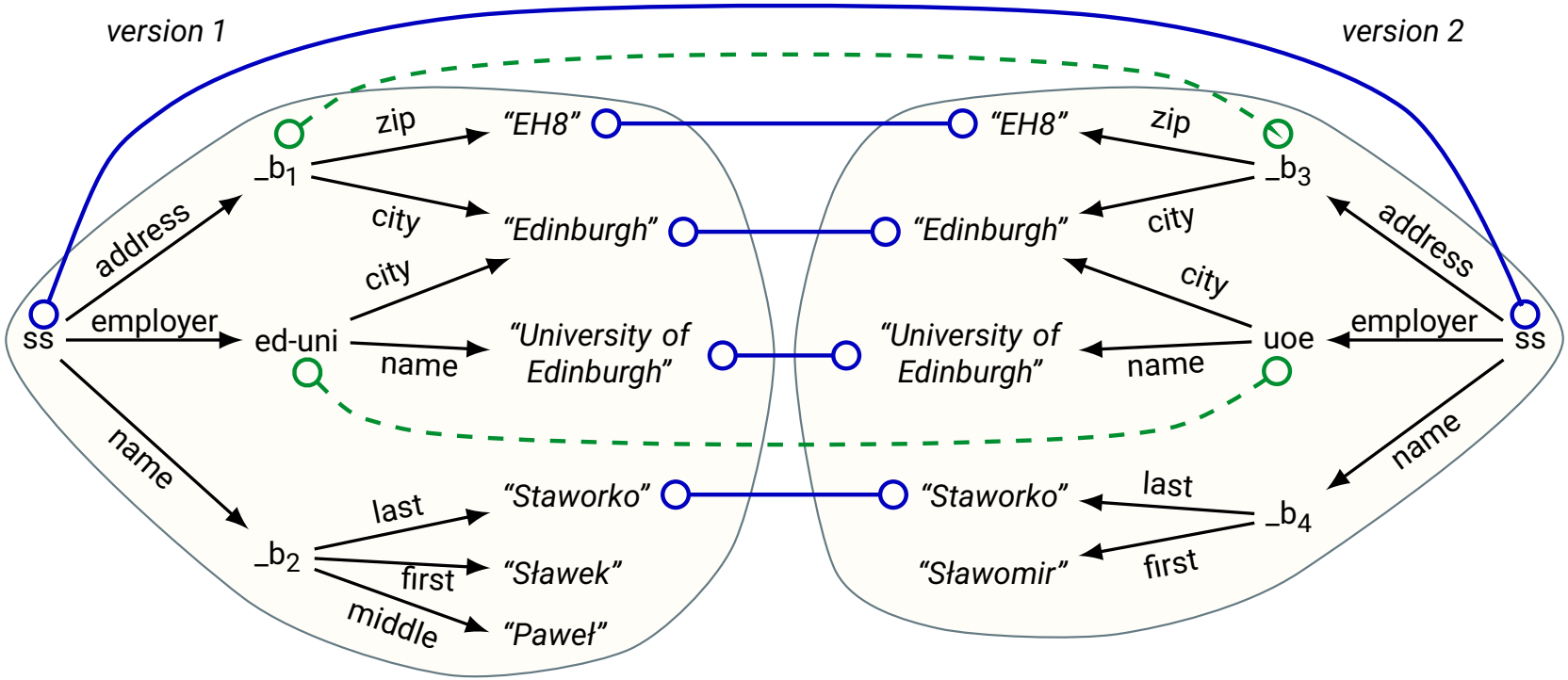
trivial alignment

label equality

Challenges

- blanks (reification, data structures)
- Changes in URI naming schemes
- Data value changes (curation)
- Graph structure modifications

Our Contributions



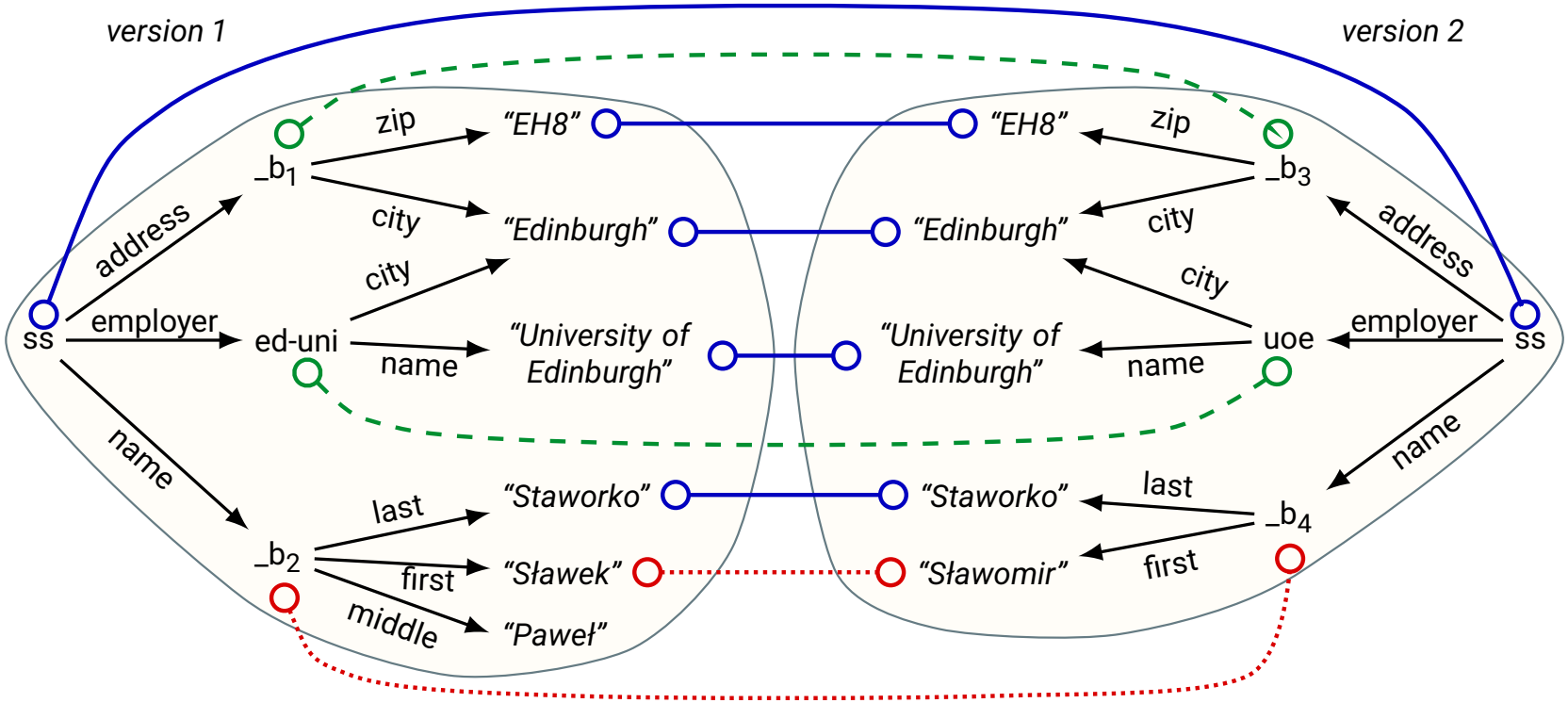
bisimulation alignment

compares node contents (outbound reachable nodes)

- *canonizes blank nodes*
- *captures ontology changes*

Bisimulation is quite efficient

Our Contributions



bisimulation alignment

compares node contents (outbound reachable nodes)

- *canonizes blank nodes*
- *captures ontology changes*

Bisimulation is quite efficient

similarity measure alignment

incorporates similarity measures (edit distance)

- *data value changes*
- *graph structure changes*

Inherent High Complexity

Overview

1. Alignment with graph colorings and bisimulation

2. Alignment with similarity measures

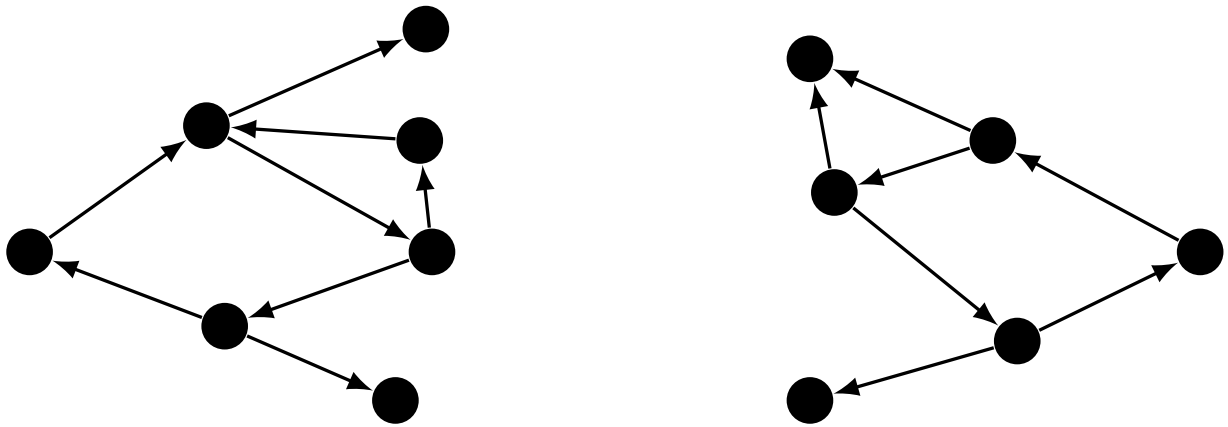
3. Weighted graph colorings and the overlap heuristic

4. Experimental evaluation

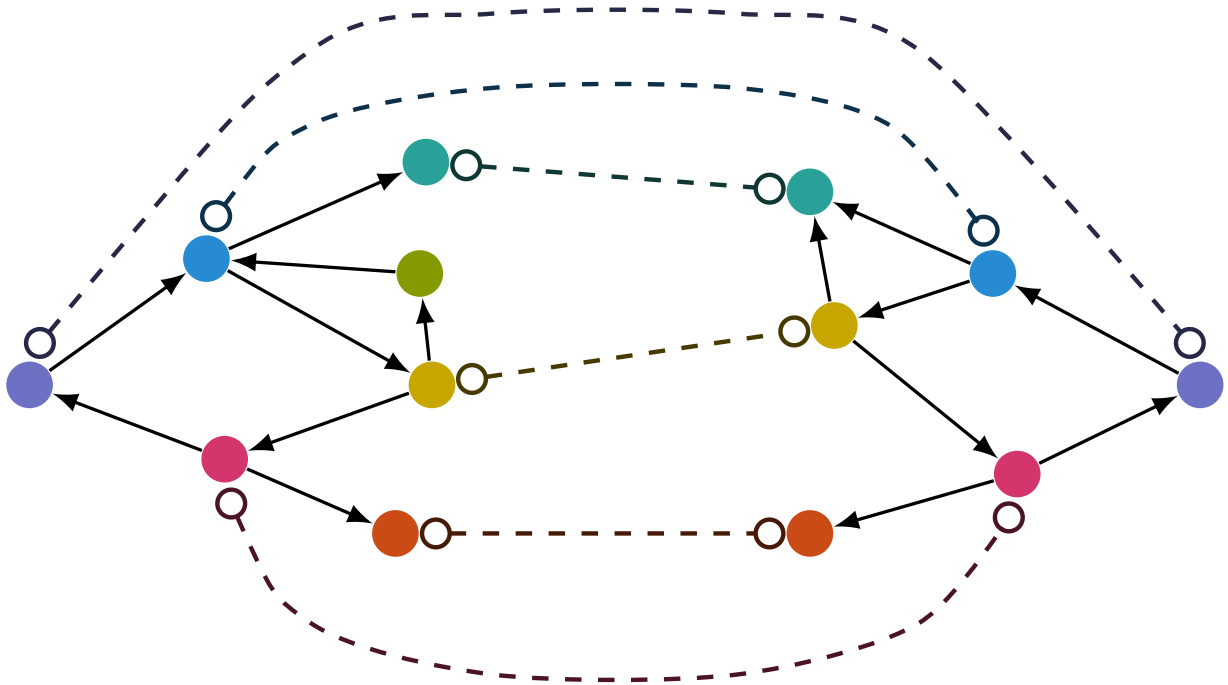
Aligning with Graph Colorings

one combined graph $=$

source graph $\uplus$ target graph



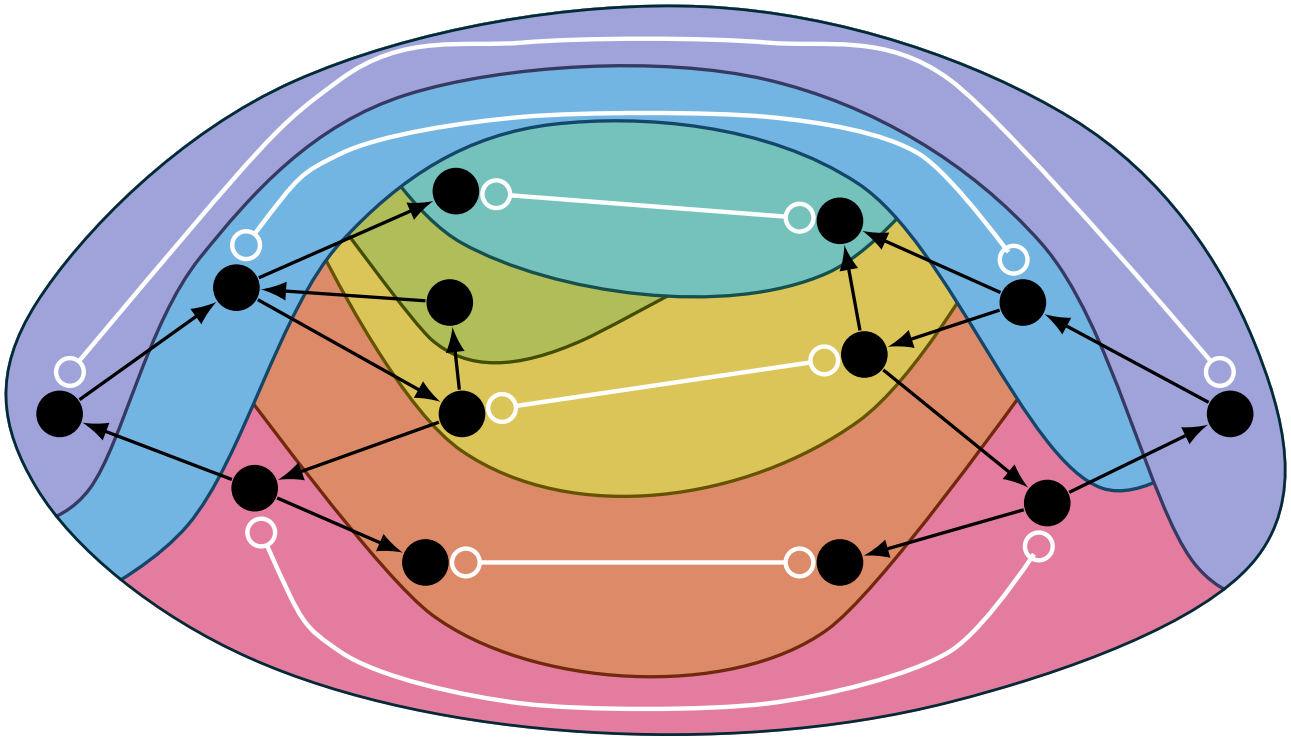
Aligning with Graph Colorings



Graph coloring maps nodes to colors (and colors can be anything, e.g. URIs)

A source node is aligned to a target node if the nodes have the same color

Aligning with Graph Colorings



Graph coloring maps nodes to colors (and colors can be anything, e.g. URIs)

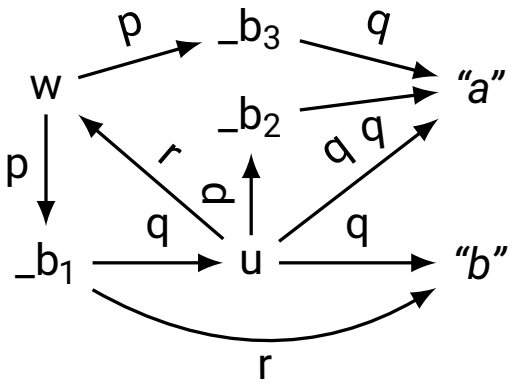
A source node is aligned to a target node if the nodes have the same color

A coloring defines a partition of the graph into clusters of nodes of the same color

A partition can also be viewed as an equivalence relation, where

two nodes are equivalent if they have the same color

Bisimulation

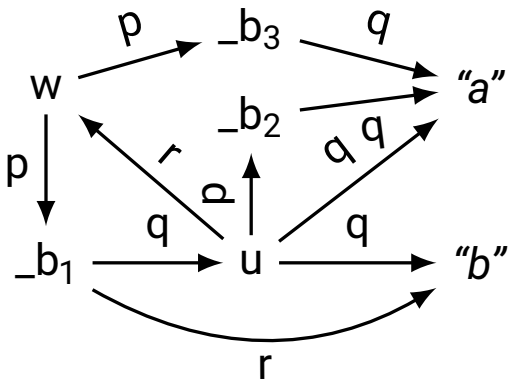


m simulates *n* iff *m* and *n* have the same label *n* and for every edge (n, p, n') there exists an edge (m, p, m') such that m' simulates m'.

n and *m* are bisimilar if *n* simulates *m* and vice versa.

Two nodes are bisimilar if they cannot be distinguished by means of tree patterns

Bisimulation



m simulates *n* iff *m* and *n* have the same label *n* and for every edge (n, p, n') there exists an edge (m, p, m') such that m' simulates m'.

n and *m* are bisimilar if *n* simulates *m* and vice versa.

Two nodes are bisimilar if they cannot be distinguished by means of tree patterns

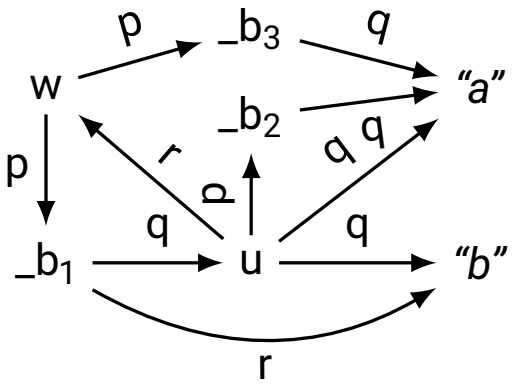
Bisimulation procedure

Iteratively refine graph coloring

1. Start with the initial coloring: Maps a node to its label
2. Refine coloring: Combine the color of a node with the color of its outbound neighborhood
3. Repeat 2 until fix-point is reached

	w	u	_b1	_b2 _b3
λ_0:	w	u	b	

Bisimulation



m simulates *n* iff *m* and *n* have the same label *n* and for every edge (n, p, n') there exists an edge (m, p, m') such that m' simulates m'.

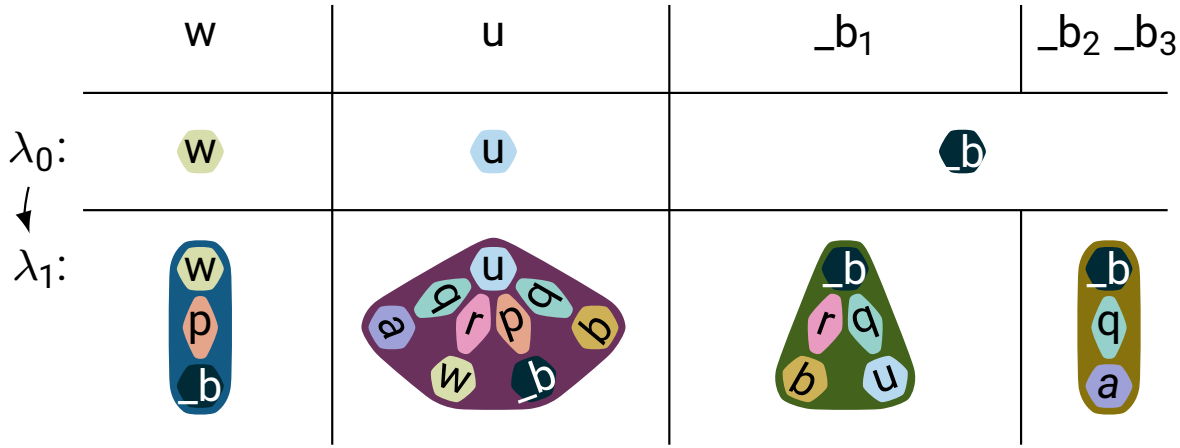
n and *m* are bisimilar if *n* simulates *m* and vice versa.

Two nodes are bisimilar if they cannot be distinguished by means of tree patterns

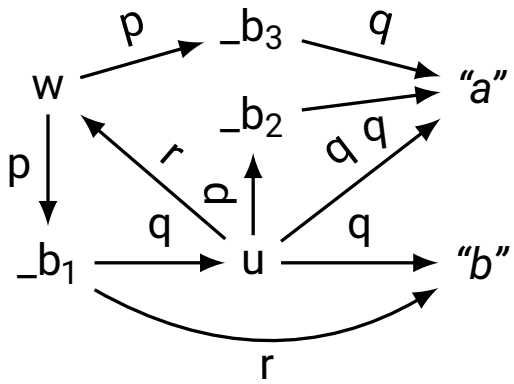
Bisimulation procedure

Iteratively refine graph coloring

1. Start with the initial coloring: Maps a node to its label
2. Refine coloring: Combine the color of a node with the color of its outbound neighborhood
3. Repeat 2 until fix-point is reached



Bisimulation



m simulates *n* iff *m* and *n* have the same label *n* and for every edge (n, p, n') there exists an edge (m, p, m') such that m' simulates m'.

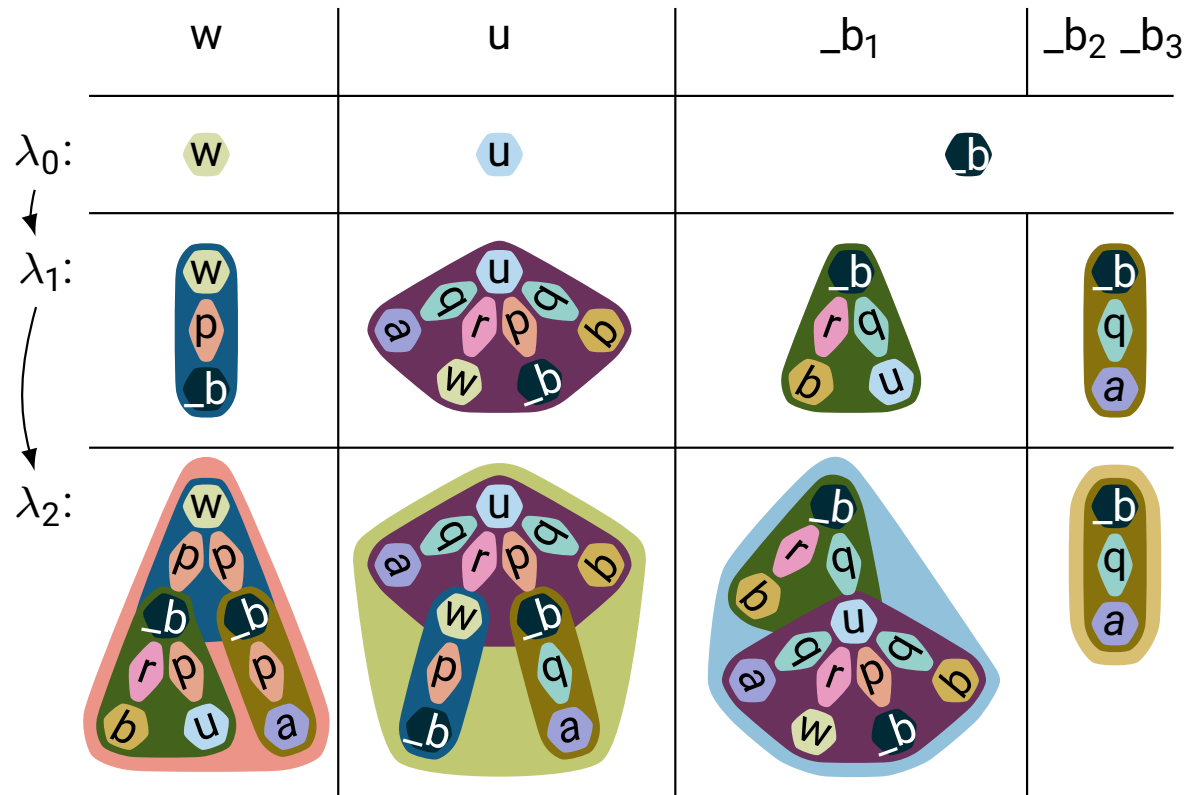
n and *m* are bisimilar if *n* simulates *m* and vice versa.

Two nodes are bisimilar if they cannot be distinguished by means of tree patterns

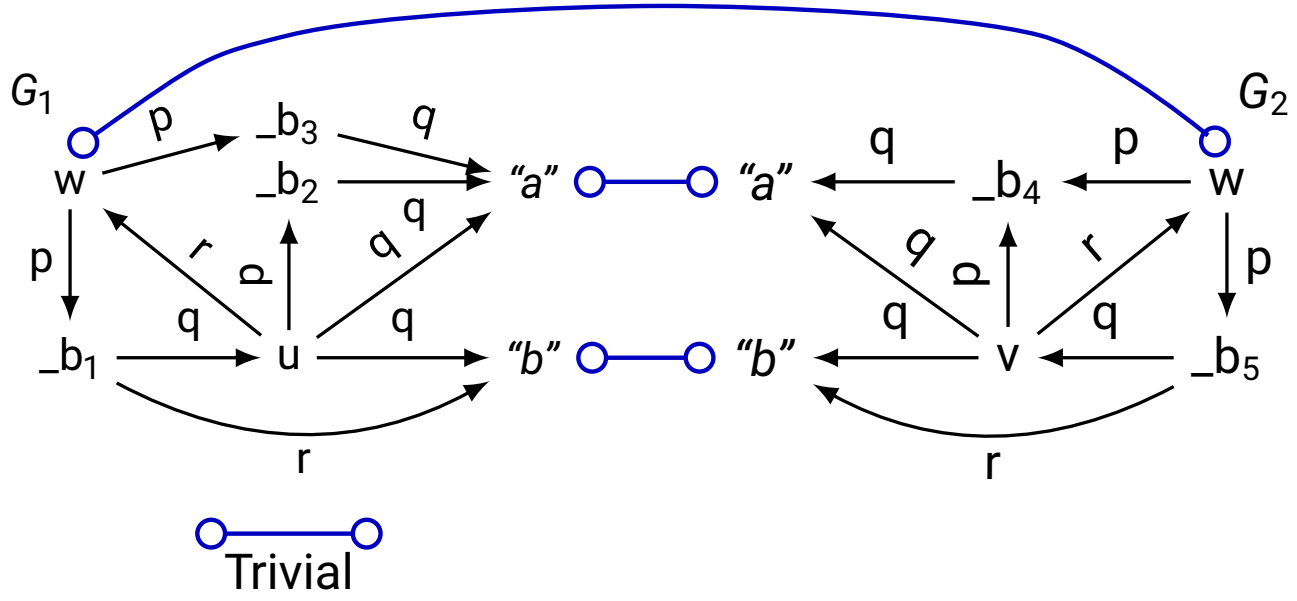
Bisimulation procedure

Iteratively refine graph coloring

1. Start with the initial coloring: Maps a node to its label
2. Refine coloring: Combine the color of a node with the color of its outbound neighborhood
3. Repeat 2 until fix-point is reached



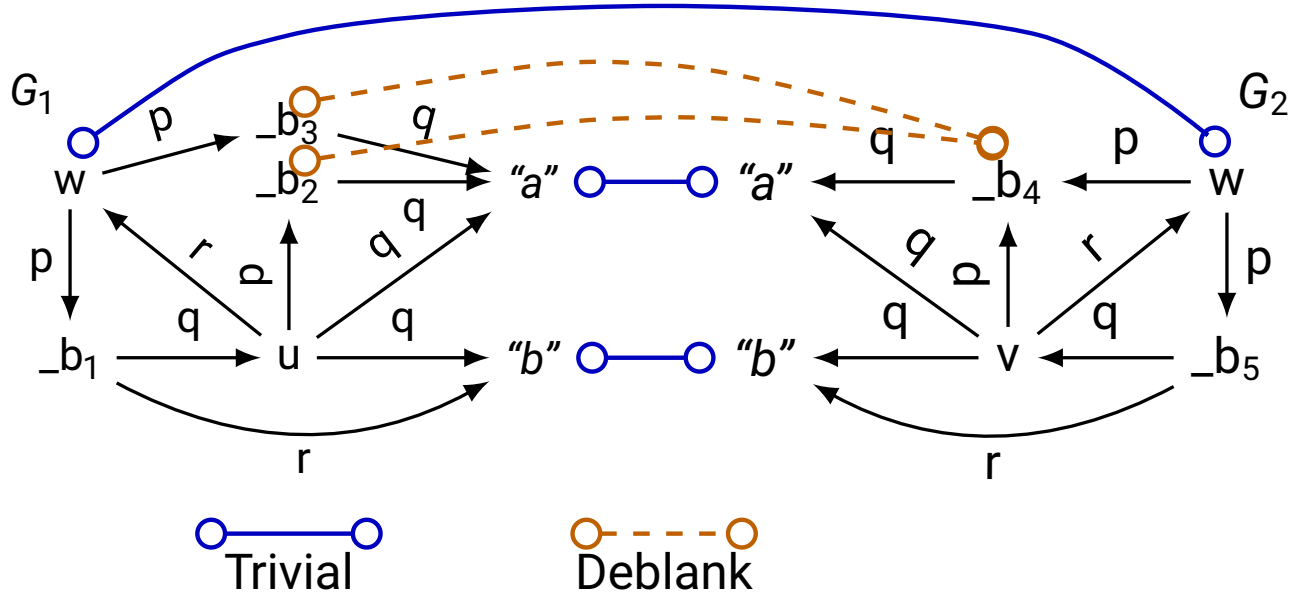
Deblanking alignment



Run bisimulation refinement procedure on blank nodes only

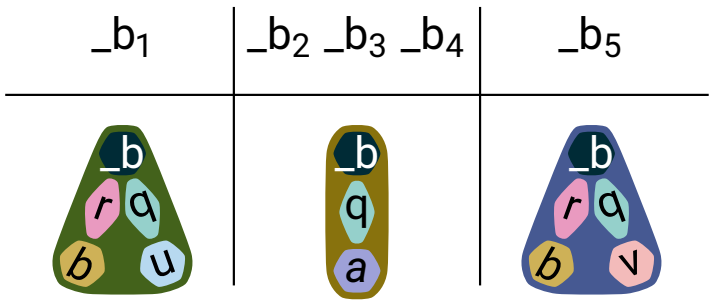
(colors of URIs and literals do not change)

Deblanking alignment

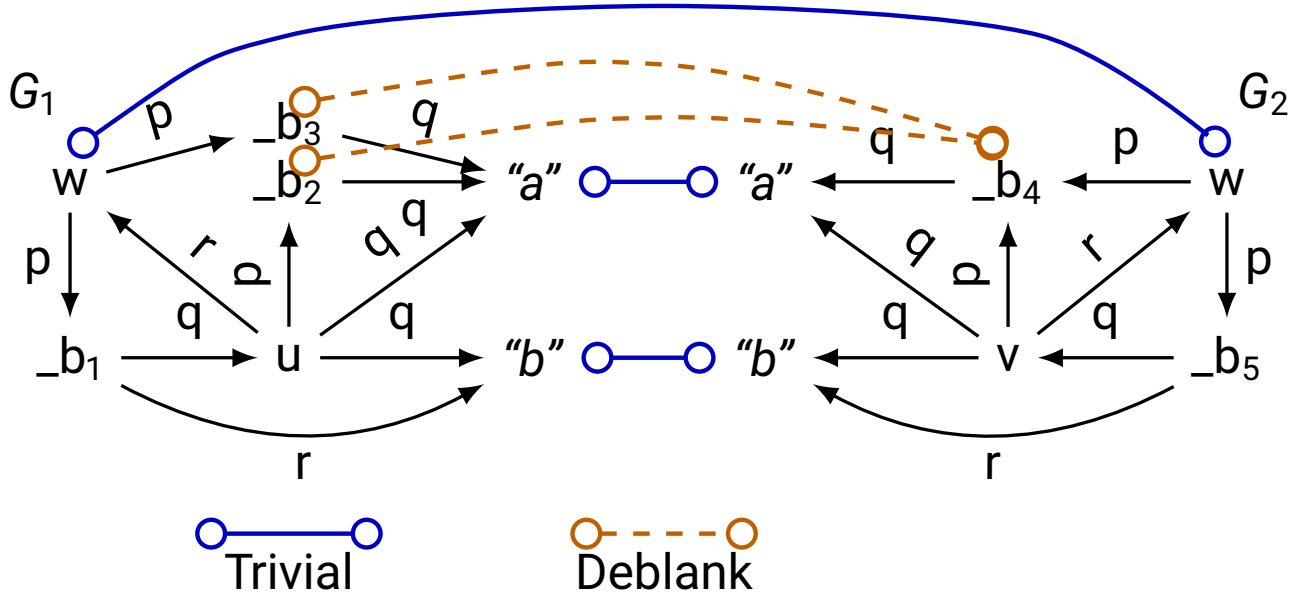


Run bisimulation refinement procedure on blank nodes only

(colors of URIs and literals do not change)

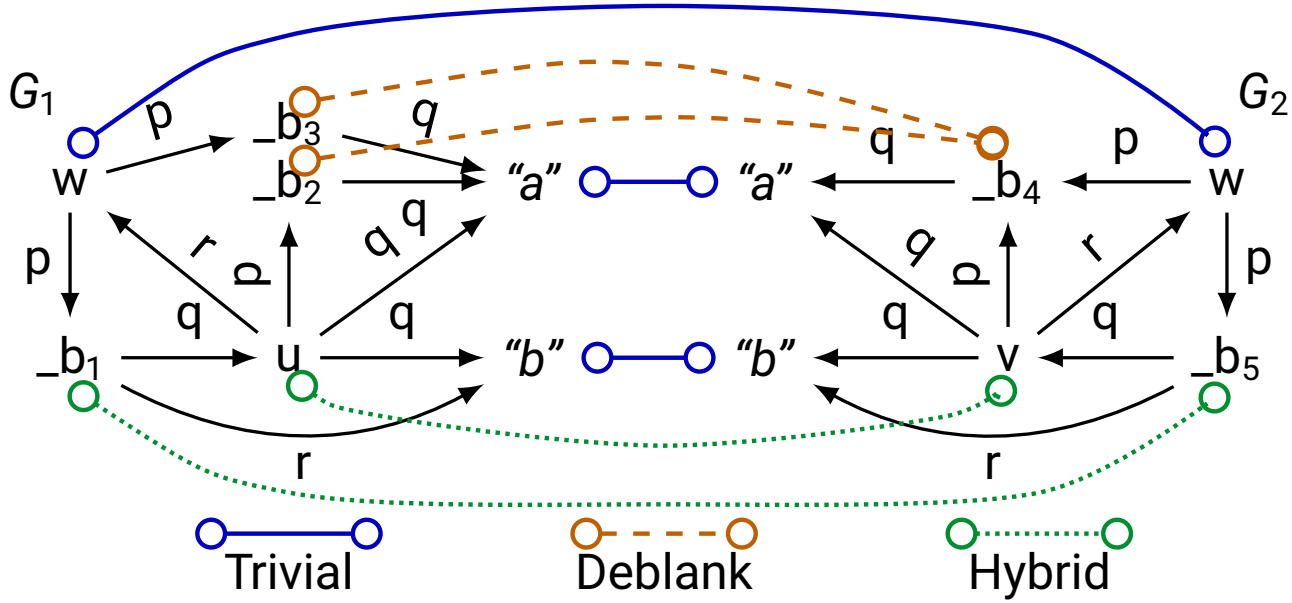


Hybrid alignment

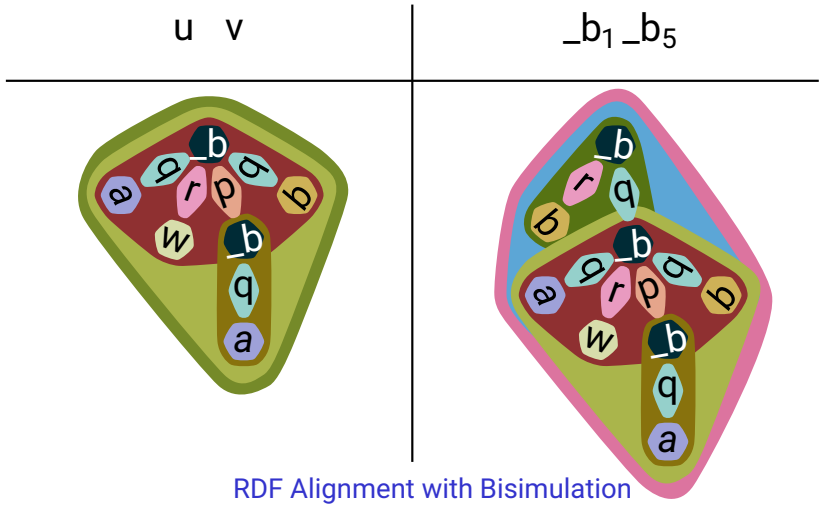


Blank out all unaligned URI nodes and apply
the bisimulation procedure on all blank nodes

Hybrid alignment



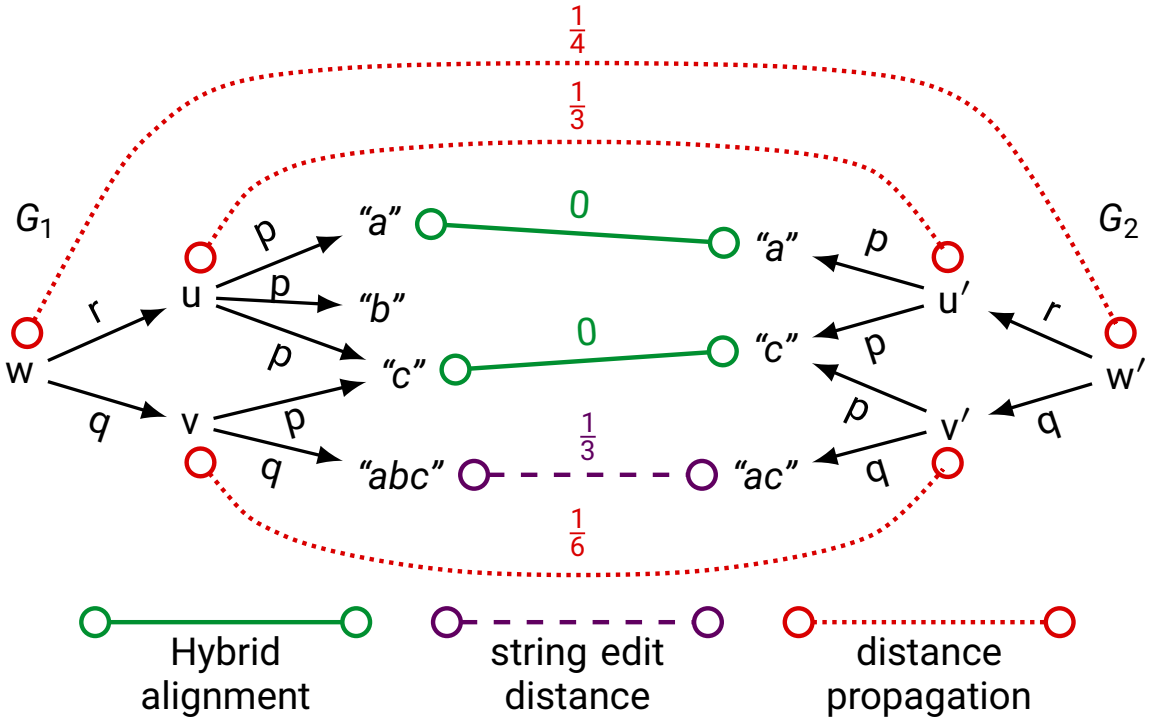
Blank out all unaligned URI nodes and apply the bisimulation procedure on all blank nodes



Alignment with similarity measures

Edit distance $\sigma_{\text{Edit}} : \mathit{Nodes} \times \mathit{Nodes} \to [0; 1]$

0 – close, similar nodes
1 – distant, dissimilar nodes

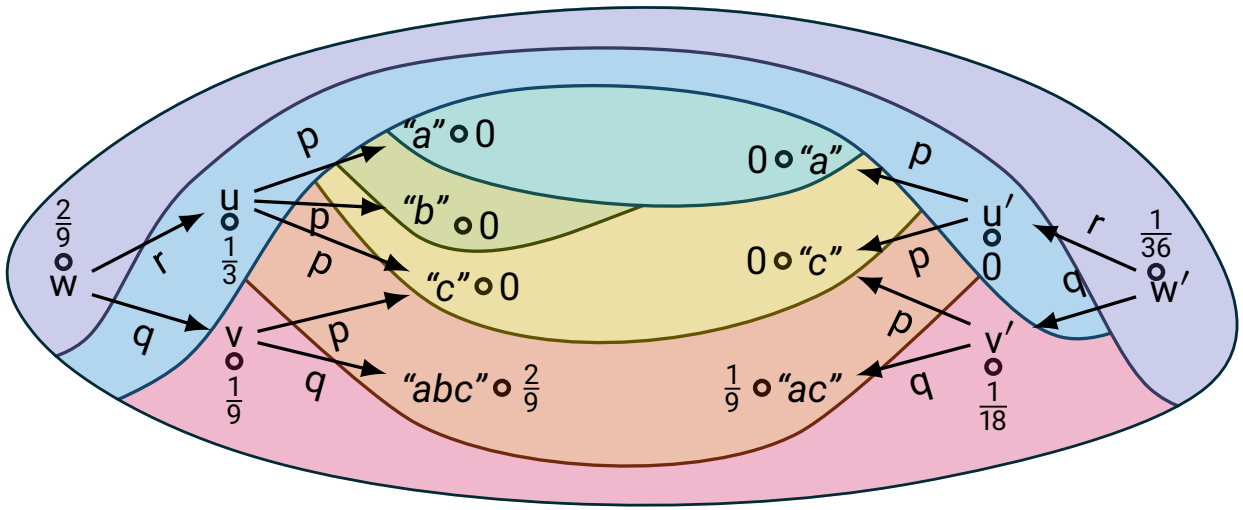


- 0 for nodes aligned by the Hybrid alignment
- standard string edit distance on literal nodes
- iterative propagation to other nodes with Hungarian algorithm in a manner robust under data changes and changes in the graph structure

Lower bounds in $O(n^2)$ – impractical for large RDF

Weighted Colorings and Overlap Heuristic

Weighted colorings additionally specify
the distance of each node from the center of its cluster.



- The weighted coloring is computed using Overlap heuristic, a refinement procedure designed to approximate the edit distance.
- Distance between two nodes from the same cluster is estimated with triangle inequality, e.g., $\sigma(v, v') = \frac{1}{9} + \frac{1}{18}$.

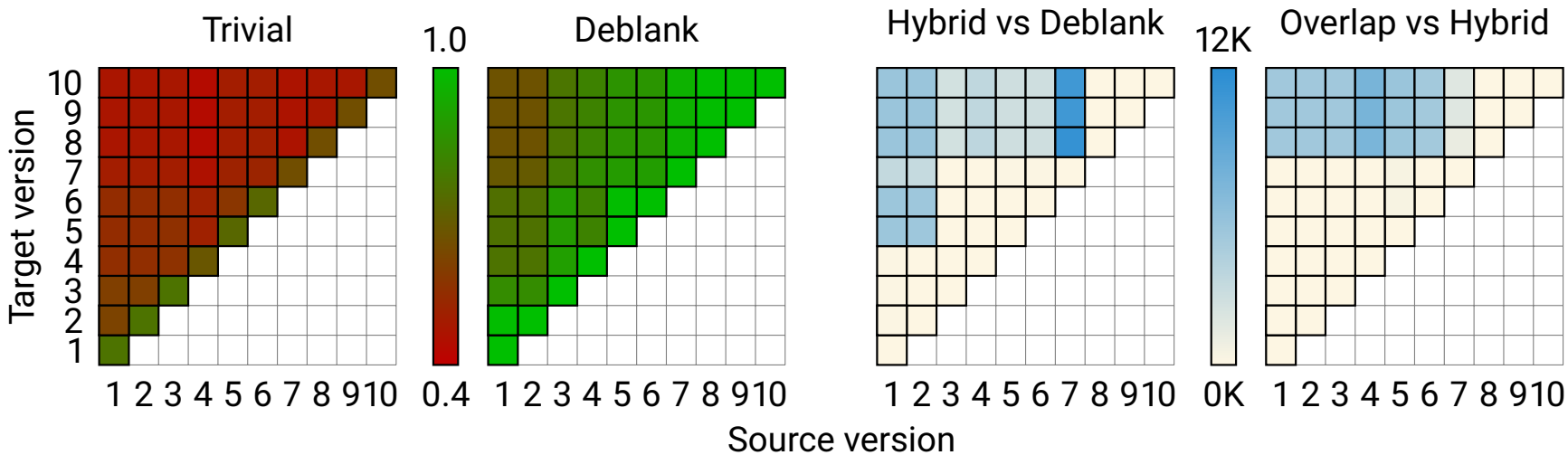
Thm. For n and m from the same cluster $\sigma_{\text{Edit}}(n, m) \leq \mathit{weight}_{\text{Overlap}}(n) + \mathit{weight}_{\text{Overlap}}(m)$.

Experimental evaluation

Experimental Factor Ontology (EFO)

Practical dataset

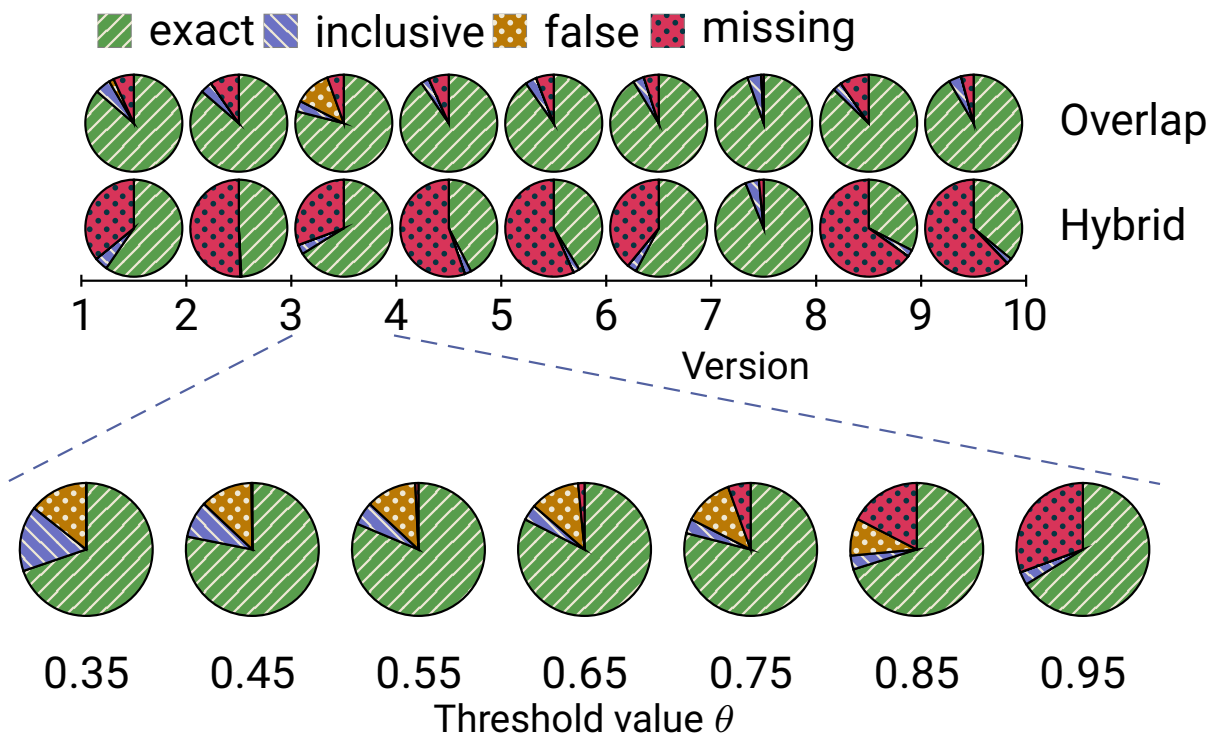
- a curated evolving OWL ontology represented in RDF
- 10 consecutive versions
- 120K-150K nodes: 10% URIs and 10-15% blanks used for reification and complex data structures; 200-250K triples
- ontology prefix changes
  (e.g., `http://purl.org/obo/owl/` → `http://purl.obolibrary.org/obo/`)



Guide to Pharmacology database (GtoPdb)

Quasi-synthetic dataset with ground truth (for precision evaluation)

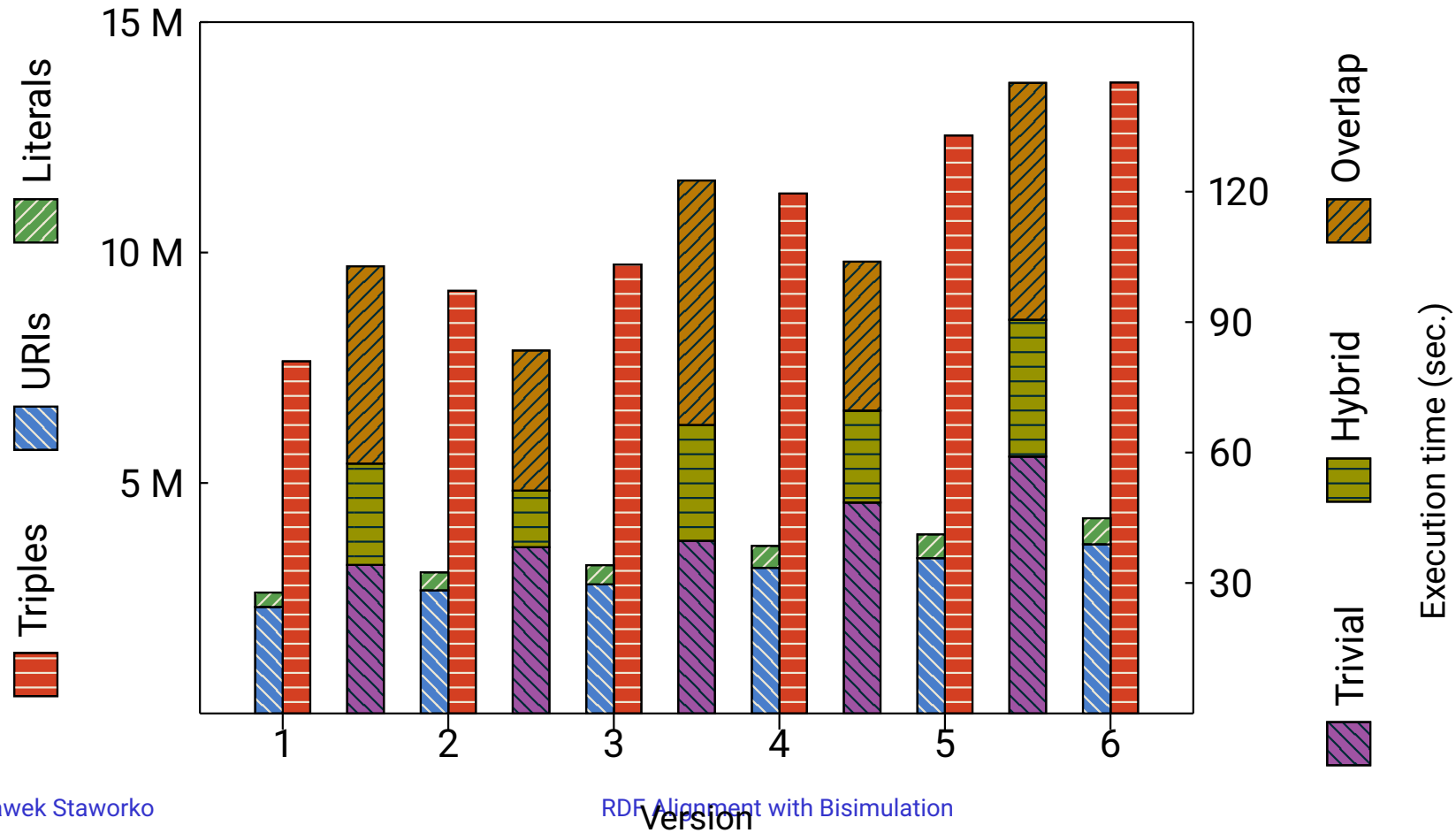
- a curated evolving relational database exported to RDF (W3C DM)
- ground truth for evaluating Hybrid and Overlap alignments
- 0.5M-1M nodes (50/50 URIs and literals); 2M-6M triples



DBpedia

A large dataset (for scalability evaluation)

- a subset of DBpedia containing category information
- 2.6-4.2M nodes and 7.6-13.7M triples.
- performance in line with the state of art for bisimulation computation



Conclusions and Future Work

A framework for aligning RDF graphs that is

- practical, effective, and scalable (for evolving RDF)
- generic (can be easily customized to more elaborate tasks)

In the future

- using notions of key for color refinement (parts of node contents)
- use not only the contents of a node but also its context
- compact representation of a timeline of a RDF database
- efficient (ShEx) schema inference