WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

Counting to *k*, or how SPARQL1.1 Property Paths Can Be Extended to Top-k Path Queries

Vadim Savenkov
based on joint work with Erwin Filtz, Jürgen Umbrich, Axel Polleres and Qaiser Mehmood

HDT Tutorial 2017, ISWC 2017 Vienna
October 20, 2017

Shortcomings of the existential sematnics of path expressions

WU WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

No way to get the nodes / properties / length of arbitrary paths in SPARQL 1.1

EFMD EQUIS ACCREDITED

Top k shortest path queries

- More informative than the existential semantics
- Avoids the double exponential blow-up of exhaustive path enumeration
- However: non-deterministic

Classic top k shortest paths algorithms

WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

Yen [Yen 71]

- Extends shortest path computation (e.g. with Dijkstra's algorithm)
- Only works for **acyclic paths**

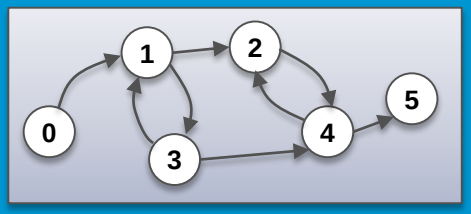
Eppstein [Eppstein 99]

- Adds support for cycles
- Memory-efficient representation of multiple paths

Main applications in routing (traffic, networks), where **weigths are essential**.

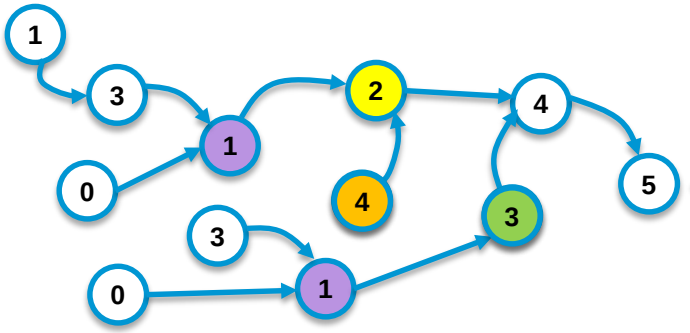
RDF graphs not weighted ⇒ unnecessary overhead.

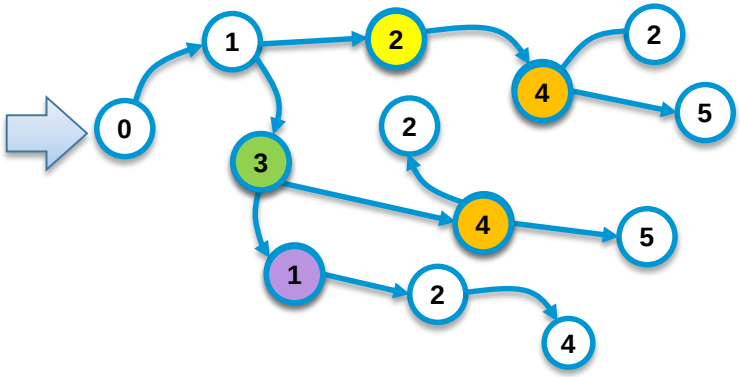
Bidirectional BFS



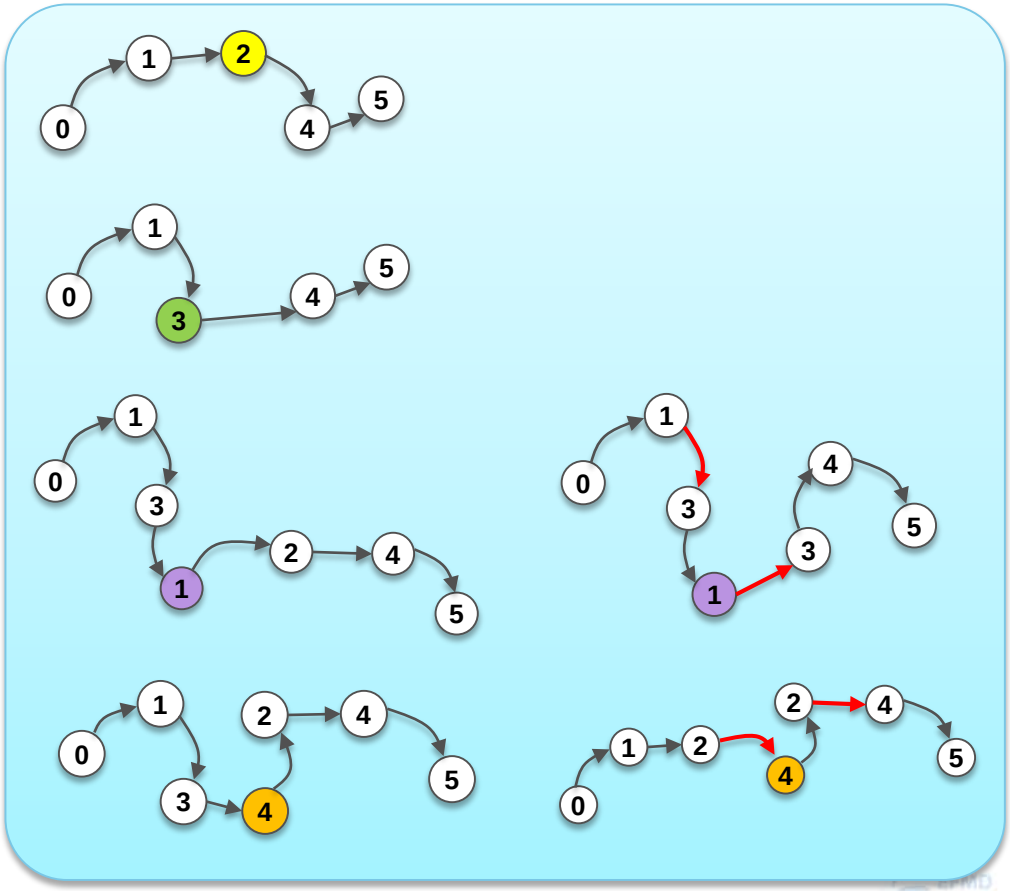
WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

Traversals





Paths found



EQUIS ACCREDITED

Native HDT Interface

```
public interface GraphIndex<V,E> {
    Iterator<Edge<V,E>> lookupEdges(V source, V target);
    PathArbiter<V,E> createAllPassArbiter();
}

public class HDTIntGraphIndex implements GraphIndex<Integer,Integer> {
    public final HDT hdt;
    public final org.rdfhdt.hdt.dictionary.Dictionary dict;
…
  @Override
   public Iterator<Edge<Integer,Integer>> lookupEdges(Integer source, Integer target){
      if(source == null){ source = 0;  }
      if(target == null){  target = 0;    }
      TripleID tripleid = new TripleID(source, 0, target);
      IteratorTripleID result = null;
      try {
         result = hdt.getTriples().search(tripleid);
      }
      catch(IndexOutOfBoundsException ex){ }
      return result!=null? new TripleIterator(result, source.equals(0)) :
            new EmptyTripleIterator();
   }
```

Our current approach

WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

- Based on bi-directional BFS
- Adds support for path expressions
- Flexible implementation due to Java generics
- Packed as an extension library (Jena ARQ), currently over HDT graphs
  - fast lookups to get (outgoing/incoming) edges
  - memory efficient due to dictionary encoding IRIs and strings as integers
- Outperforms similar approaches as well as existential path queries in established commercial systems (Stardog, Blazegraph,Virtuoso)

EFMD EQUIS ACCREDITED

hdt-jena based interface

```
public class RDFGraphIndex implements GraphIndex<Node,Node> {
   …
   Graph graph;

   public RDFGraphIndex(Graph graph){
      this.graph = graph;
   }

   @Override
   public Iterator<Edge<Node, Node>> lookupEdges(Node source, Node target) {
      return new WrappingIterator(graph.find(source,null,target), source==null);
   }
```

+ Some workarouds

WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

```
@Override
public void init(GraphIndex<Integer,Integer> gi, Integer source, Integer target, boolean bidirectional)
{
    hdt = (HDTIntGraphIndex)gi;
    if( hdt!=null ){
        this.target = target;
        this.nShared = hdt.dict.getNshared();
    }
}

@Override
public CumulativeRank rankEdge(Edge<Integer,Integer> edge, Iterable<Edge<Integer,Integer>> path,
                               CumulativeRank pathRank, boolean forkRankObject, boolean backwardPath ){

    //A workaround for the problem that literals and IRIs can have clashing HDT integer representation.
    if( hdt!=null && !backwardPath ){
        Integer v = edge.vertex();
        if( v>=nShared && v!=target ) {
            if ( v != hdt.vertexKey(hdt.vertexEntry(v, Edge.Component.TARGET), Edge.Component.SOURCE) ){
                return pruneRank;
            }
        }
    }
    return passRank;
}
```

Adding SPARQL 1.1 Property Path Expressions to BFS

- Use dk.brics.automaton [Møller17] Java library to construct an NFA based on the path expression
- Making use of Jena ARQ path expression parser
- Encode properties as characters: a / (b/c)* /d for

  foaf:knows / (dbp:predecessor/dbp:firstWin)* / dbp:parent
- For the backward search, recursively invert all sequences:

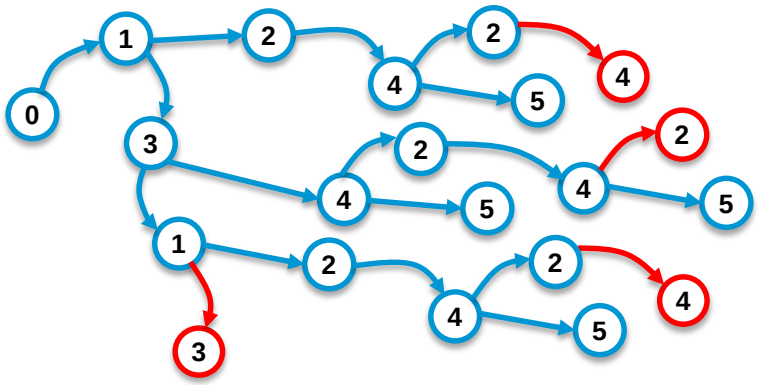
  a / (b/c)* /d ⇒ d / (c/b)* /a
- Inverse paths (e.g.: ^dbp:predecessor) currently unsupported

Path expressions

At evaluation time:

- Same IRI-to-character dictionary + special fresh character for all properties not metnioned in the expression

- Each node in the unfolding tree contatins the NFA state for early path prunning



Syntax

- Use Jena ARQ property functions (allowing for side-effects, e.g. querying the graph store and modifying the number of results)

```
PREFIX : <http://dbpedia.org/property/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX ppf: <java:at.ac.wu.arqext.path.>
SELECT ?path WHERE {
    ?path ppf:topk (dbr:Felipe_Massa dbr:Red_Bull 10
    ":firstWin/((!:)*)|(!:)*/:firstWin") }
```

- `(!:)` is a wildcard (negation of a non-existant empty IRI)
- ?path contains paths represented as concatenated strings
- **Source and target nodes need to be bound in the query**

Evaluation Setup

WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

Based on DBpedia SPARQL Benchmark [Morsey et al. 11], as used in the ESWC 2016 Top k Shortest Paths Challenge [Papadakis et al. 16]

	Triples	Subjects	Pred's	Objects	Shared
0.1DB	9 264 609	313 036	13 114	3 482 820	58 535
1DB	46 275 619	1 457 983	21 875	13 751 780	462 478

Queries (with k varying from 3 to 400K)

Optional restriction on the first OR the last edge

ID	Source node (dbr:)	Target node (dbr:)	Property (dbp:)
Q1	Felipe_Massa	Red_Bull	firstWin
Q2	1952_Winter_Olympics	Elliot_Richardson	after
Q3	Karl_W._Hofmann	Elliot_Richardson	predecessor
Q4	Karl_K._Polk	Felix_Grundy	president

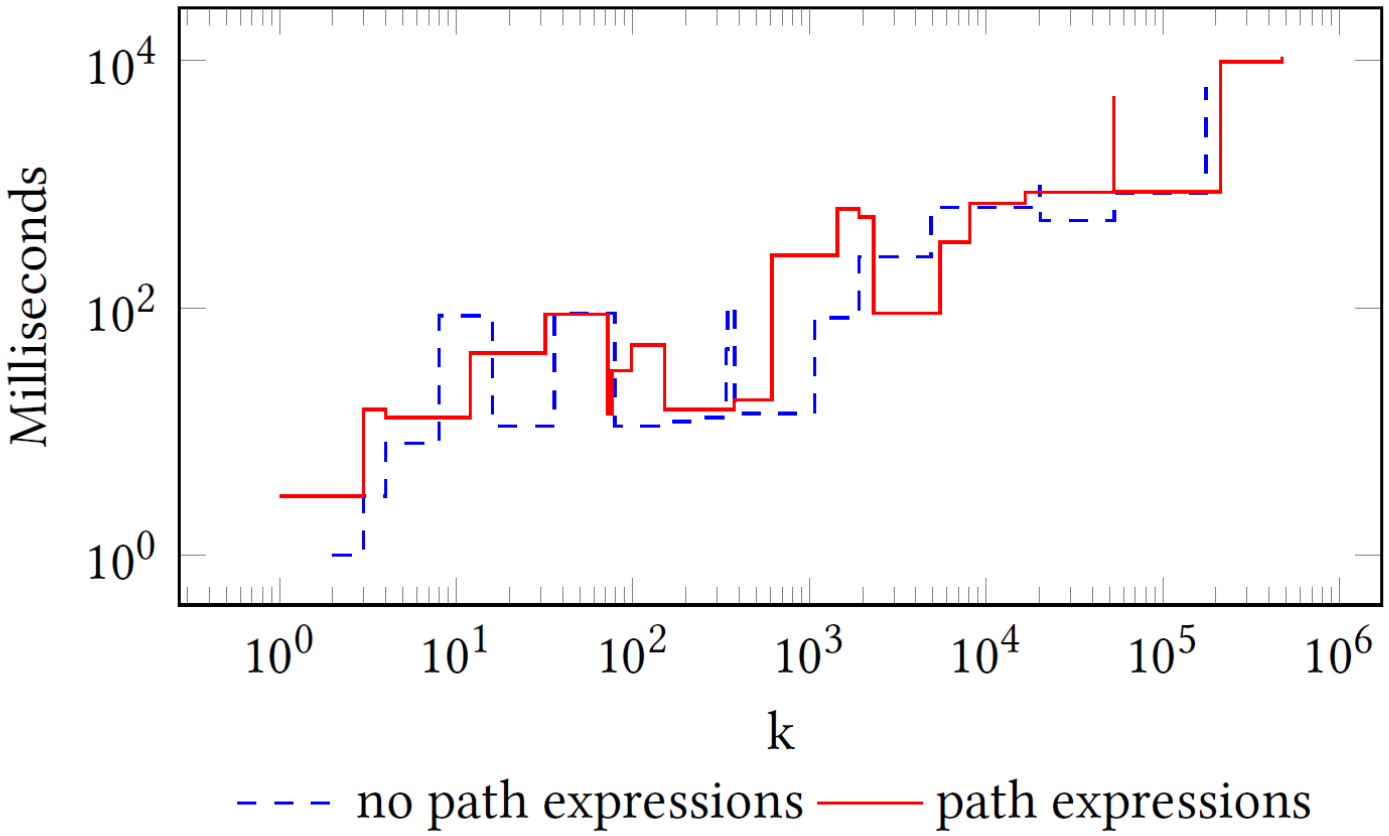
Comparing against Stardog

Query	$k = 1$	$k = 100$	Stardog (reachability)
Q1 (!:*)	24	94	57
Q1	125	6 138	19,732
Q2 (!:*)	15	132	45
Q2	19	2 568	34 707

Performance of top k / reachability queries

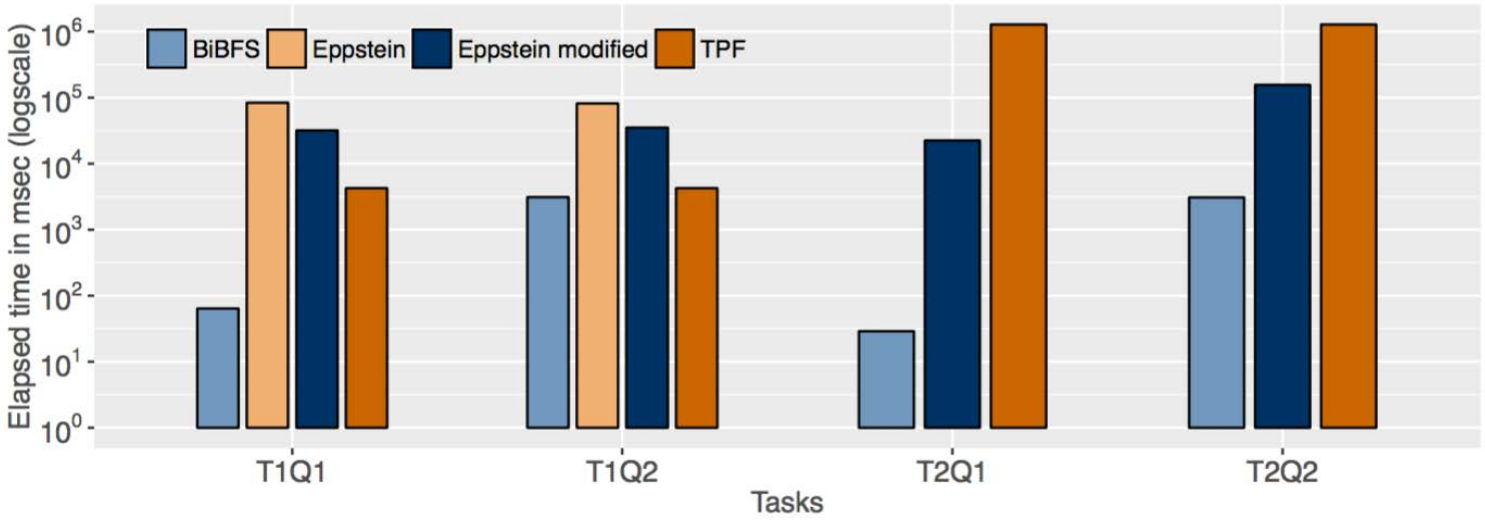
Blazegraph and Virtuoso did not deliver results (no results, or memory limits hit).

Path expressions support is inexpensive



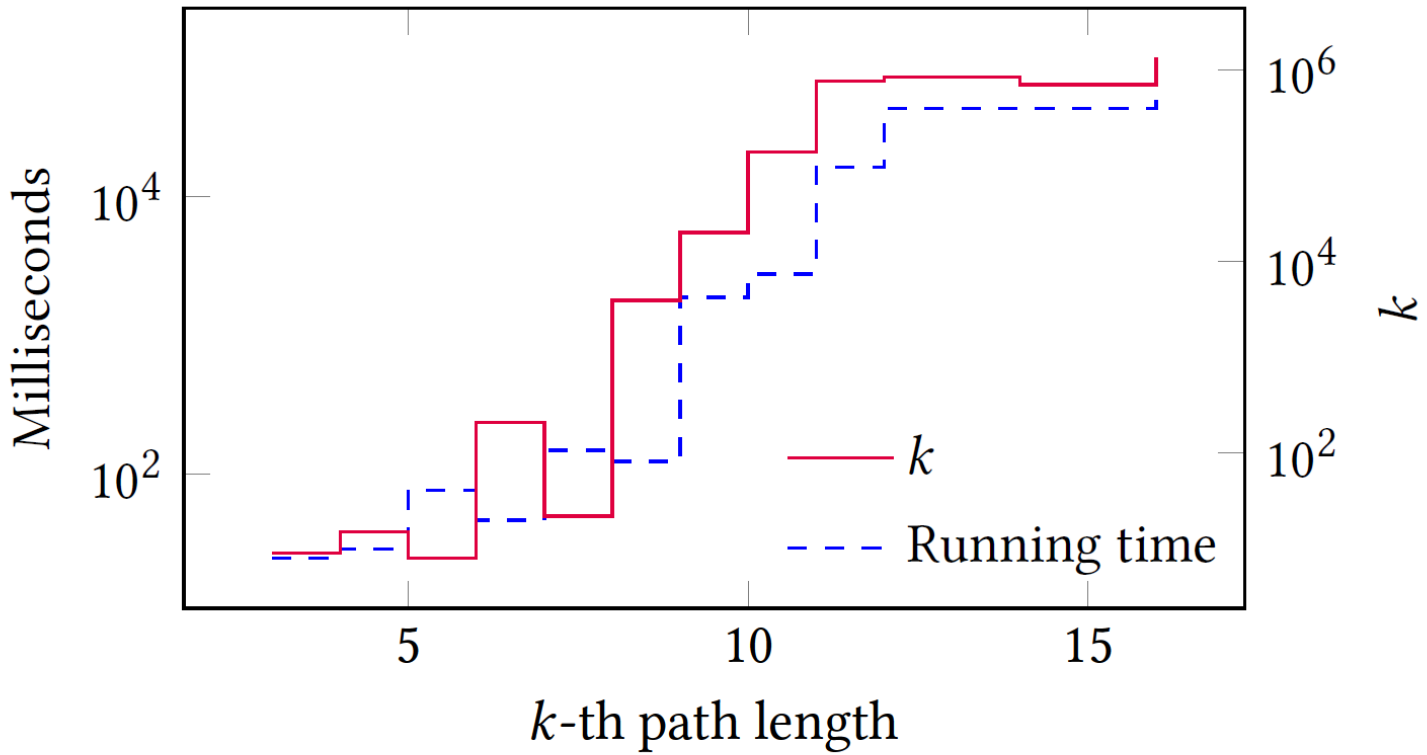
Comparison against the ESWC 2016 challenge finalists

- **Eppstein**: dataset modification: no multi-edges, all weights =1
- Reported runtimes from
  - [Hertling et al.16] (**Eppstein modified**)
  - [De Vocht et al.16] (**TPF)** Triple pattern fragments



Experiment: Random paths

- Random walk to ensure connectivity with a given path length
- Increase k to ensure that the desired path length is hit



Conclusion

- Memory intensive BFS can be adapted to find top-k shortest paths in RDF graphs using HDT compression.
- Path expressions do not incur significant costs.
- Simple and easily extensible approach.

Future Work

- Support more SPARQL engines via appropriate extension interfaces
- Support special indexes for reachability.
- Paths in federated queries?

Thank you!

This work was funded by the following projects:

Project Graphsense was funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under the program "ICT of the Future", between September 2015 and November 2017.

Project Open Data for Local Communities was funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under the program "ICT of the Future", between November 2016 and April 2019.

More information https://iktderzukunft.at/en/

Qaiser Mehmood was supported by a research grant from Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289.

Literature

[Arenas et al.12] Marcelo Arenas, Sebastián Conca, Jorge Pérez: Counting beyond a Yottabyte, or how SPARQL 1.1 property paths will prevent adoption of the standard. WWW 2012: 629-638

[Eppstein99] Eppstein, D.: Finding the k shortest paths. SIAM J. Comput. 28(2), Feb 1999

[FiltzSU16] On finding the k shortest paths in RDF data, IESD (ISWC Workshop) 2016

[Hertling et al.16] Hertling, S., Schröder, M., Jilek, C., Dengel, A.: Top-k Shortest Paths in Directed Labeled Multigraphs. ESWC 2016 Challenges, Springer

[JimènezM03] Jimènez, V.M., Marzal, A.: Experimental and Efficient Algorithms: Second International Workshop, WEA 2003,

[Morsey et al. 11] Mohamed Morsey, Jens Lehmann, Sören Auer, Axel-Cyrille Ngonga Ngomo: DBpedia SPARQL Benchmark - Performance Assessment with Real Queries on Real Data. International Semantic Web Conference (1) 2011: 454-469

[Møller17] Anders Møller: dk.brics.automaton – Finite-State Automata and Regular Expressions for Java, http://www.brics.dk/automaton/

[Papadakis et al. 16] I. Papadakis, M. Stefanidakis, Ph. Mylonas, B. Endres-Niggemeyer, S. Kazanas: Top-K Shortest Paths in Large Typed RDF Datasets Challenge. 191-199

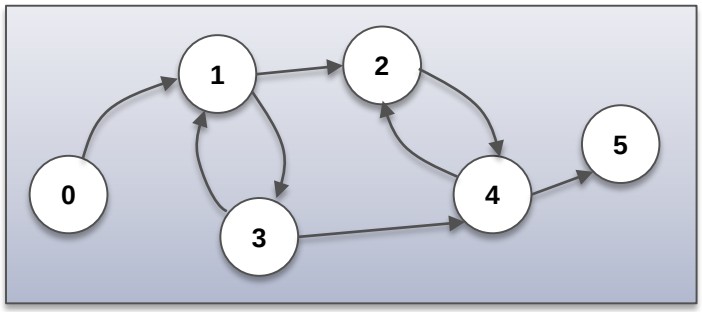
Literature [2]

[De Vocht et al.16] De Vocht, L., Verborgh, R., Mannens, E., Van de Walle, R.: Using Triple Pattern Fragments to Enable Streaming of Top-k Shortest Paths via the Web. ESWC 2016 Challenges

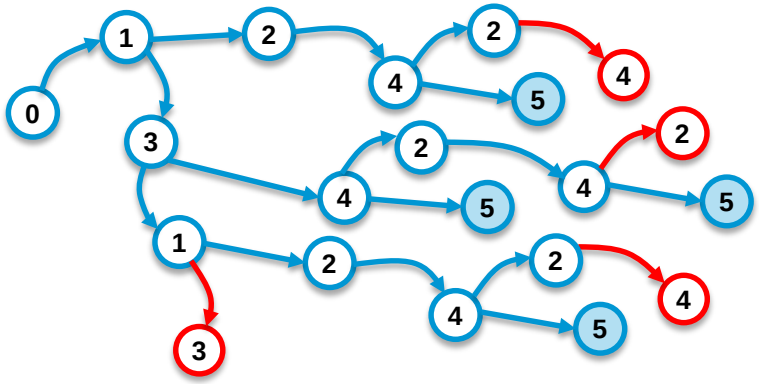
[Yen 71] J. Y. Yen, "Finding the K Shortest Loopless Paths in a Network," Management Science, 17 (11), pp. 712–716, 1971.

BFS for shortest path enumeration

- **Unfolding** the graph into a tree starting with a given node



Assuming all properties be the same in this example



Disallow exactly repeated edges (n_1, p, n_2) in paths.

That is, enumerate *trails*

The tree allows for reuse of common path prefixes