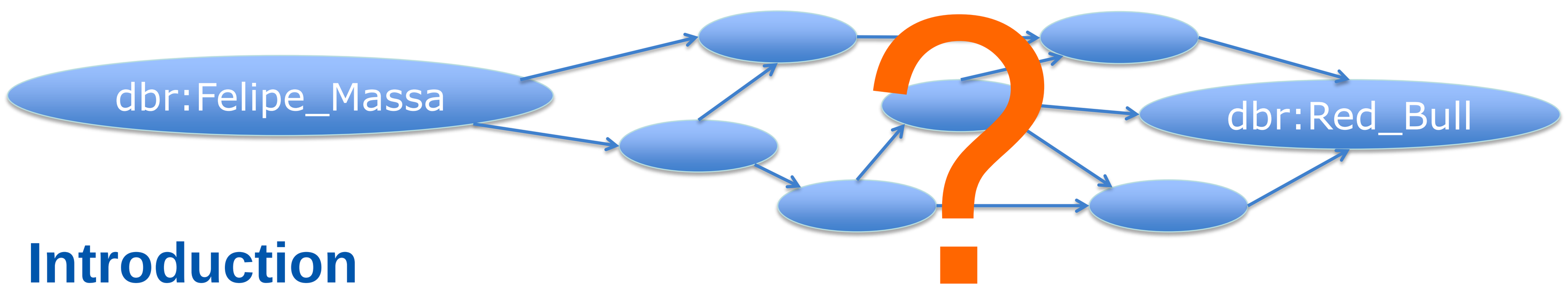
Top-k Shortest Paths in Directed Labeled Multigraphs

Sven Hertling, Markus Schröder, Christian Jilek, Andreas Dengel

German Research Center for Artificial Intelligence

Abstract

A top-k shortest path algorithm finds the k shortest paths of a given graph ordered by length. Interpreting graphs as RDF may lead to additional constraints, such as special loop restrictions or path patterns. Thus, traditional algorithms such as the ones by Dijkstra, Yen or Eppstein cannot be applied without further ado. We therefore implemented a solution method based on Eppstein's algorithm which is thoroughly discussed in this paper. Using this method we were able to solve all tasks of the ESWC 2016 Top-k Shortest Path Challenge while achieving only moderate overhead compared to the original version. However, we also identified some potential for improvements. Additionally, a concept for embedding our algorithm into a SPARQL endpoint is provided.

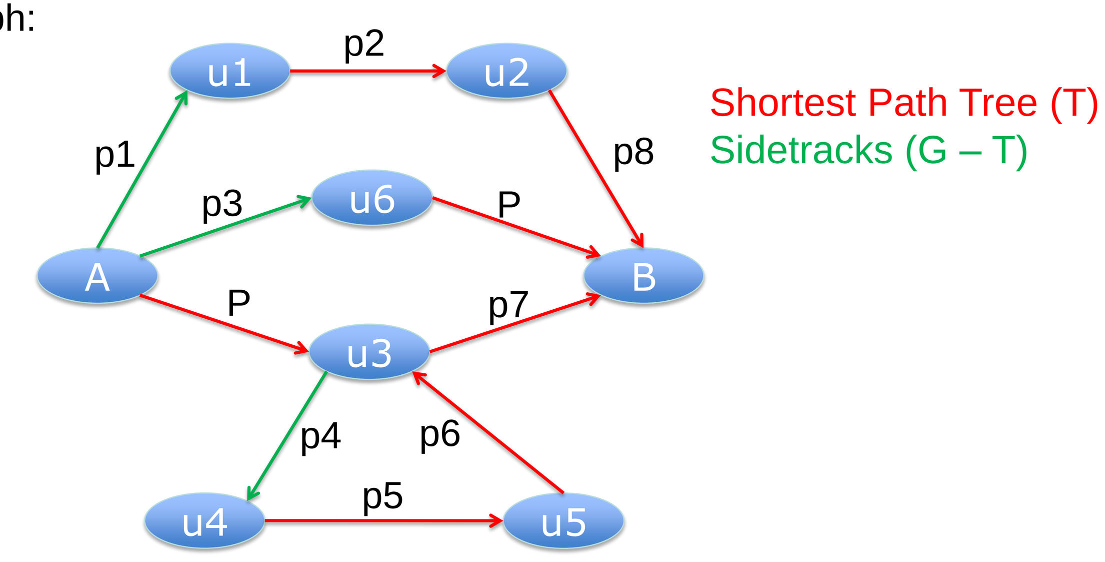


Introduction

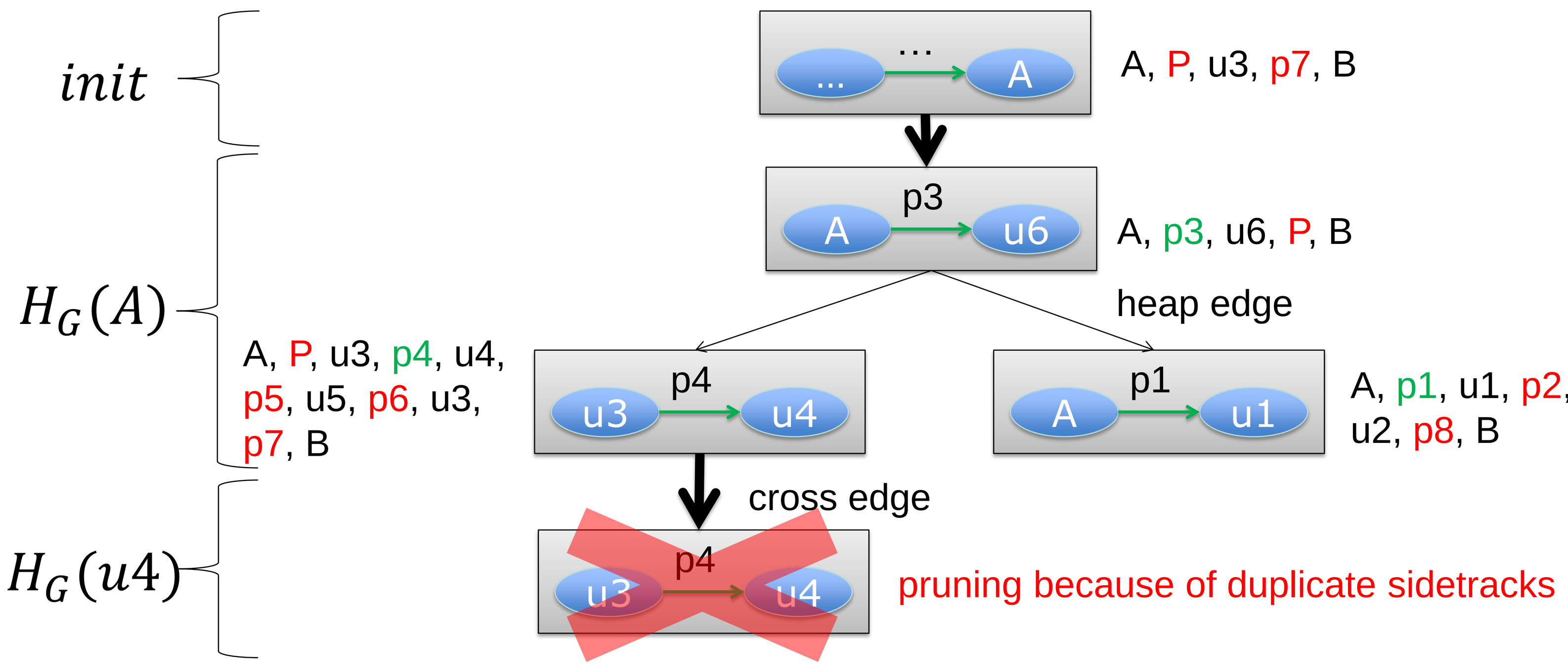
- shortest path: obvious link between resources
- subsequent shortest paths: maybe even more interesting relationships
- interpreting graphs as RDF may lead to additional constraints
- common algorithms (e.g. Dijkstra, Yen, Eppstein, …) cannot be applied without further ado

Approach

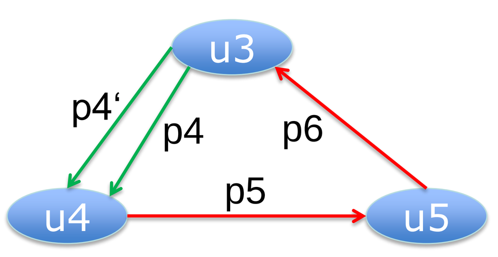
- two tasks in ESWC top-k shortest path challenge:
  1. top-k shortest valid paths
  2. every path should have the edge P as the outgoing edge of src or the incoming edge of dst
- a path is valid only if it contains unique triples (do not use the same triple twice in one path)
- vertices can be visited multiple times (Yen's algorithm which forbids loops would not find such paths)
- approach based on Eppstein algorithm (k shortest path algorithm with loops)
  - compute single destination shortest path tree T (e.g. Dijkstra) – all other edges are called sidetracks (G – T)
  - build a graph P(G) based on heaps $H_G(v)$ which orders all sidestracks on the shortest path from v to the destination
  - most vertices in P(G) correspond to a valid path (this results in a moderate overhead compared to the Eppstein algoritm)
  - a path is built up by using all activated sidetracks, otherwise use shortest path
- example graph:

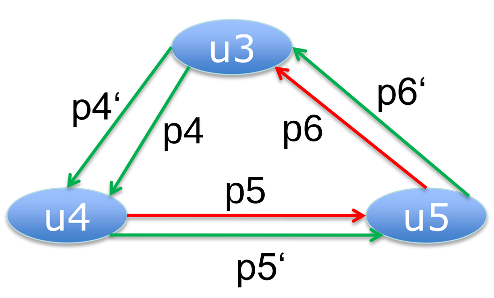


- corresponding graph P(G):



- special cases for multiple edges and cycles:





- in this case p4 and p4' are activated. Thus, triple (u4,p5,u5) is used twice. In order to detect this invalidity the path has to be built.
- in this case p4 and p4' are activated. But P(G) has to be extended since p5' and p6' can be activated in the future which would result in a valid path.

- the resulting modified Eppstein algorithm is implemented as follows:

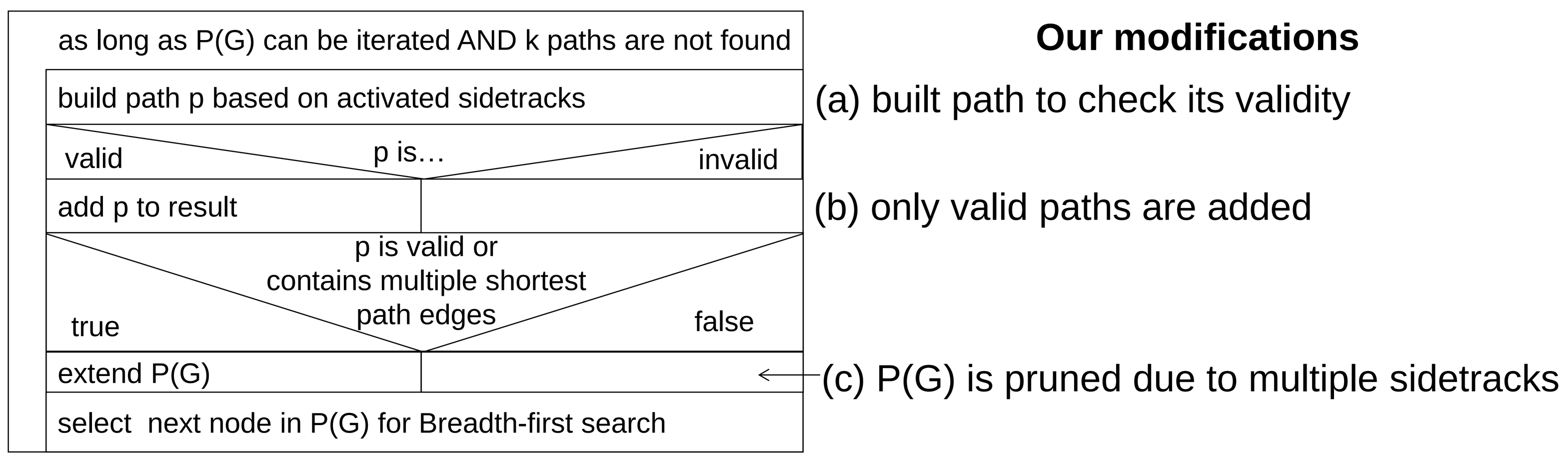
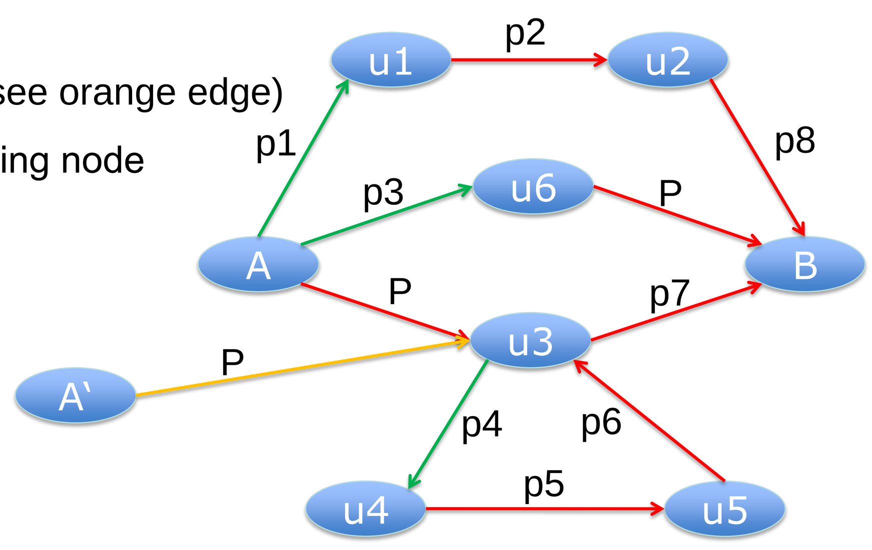


Fig. 1. Modified Eppstein Algorithm

- task two is solved in the following way:
  - a new dummy vertex A' is created
  - all P-edges starting in A are duplicated (see orange edge)
  - run modified Eppstein with A' as the starting node
  - for all paths ending on P, every edge is inverted and the above procedure is executed
  - both results are merged



Evaluation

- evaluation based on the training set of the ESWC 2016 Top-k Shortest Path Challenge
- successfully solved all queries in the challenge
- training set: 9,996,907 triples, 7,598,913 of them being literal statements, 394,085 multiple edges, 407 reflexive edges, 181,702 duplicate statements, average out degree of vertices 6.03 (s.d. 4.51)
- high k results in a high overhead since more potentially invalid paths are found
- the relation between k and the overhead of all queries is depicted in Figure 2
- algorithm's overhead increases with the number of loops

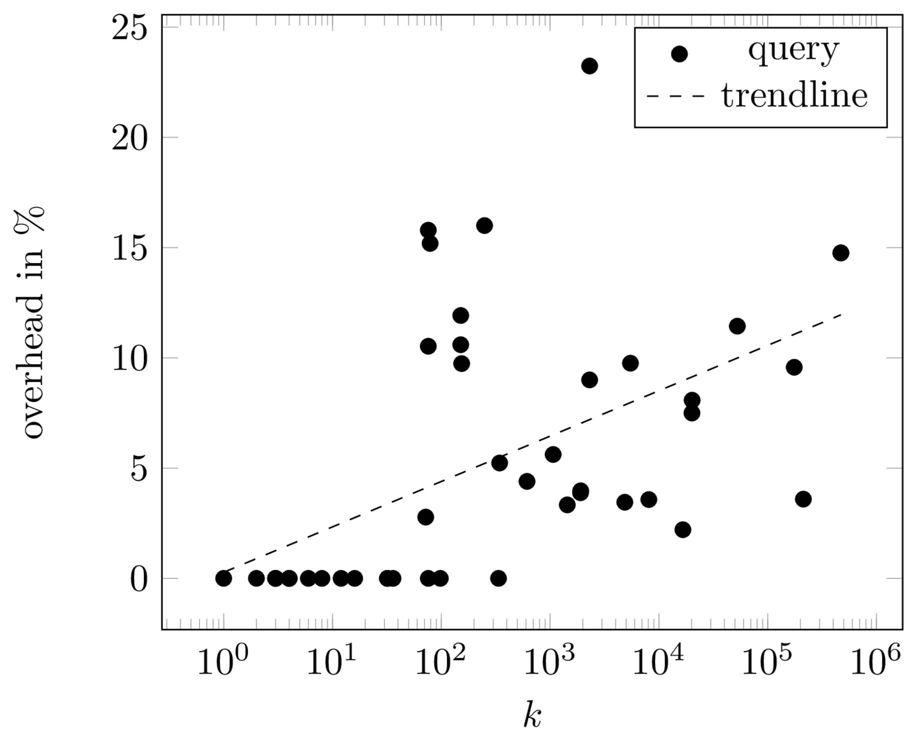


Fig. 2. Relation of k and corresponding overhead

SPARQL

- SPARQL endpoint (FUSEKI) which can extract top k shortest paths
- example query:

```
SELECT *
WHERE {
    dbr:Felipe_Massa !:* dbr:Red_Bull.
    FILTER(?r1 = dbp:after).
}
LIMIT 2
```

A !:* B = zero or more properties not matching URI „:“

- example response:

?length	?r0	?r1	?r2	?r3	?r4	?r5	?r6
7	dbr:Felipe_Massa	dbp:after	dbr:Robert_Kubica	dbp:first Win	dbr:2008_Canadian_Grand_Prix	dbp:third Team	dbr:Red_Bull
7	dbr:Felipe_Massa	dbp:after	dbr:Robert_Kubica	dbo:first Win	dbr:2008_Canadian_Grand_Prix	dbp:third Team	dbr:Red_Bull

Conclusion

- approach for finding top-k shortest paths based on Eppstein's algorithm which induces only moderate overhead
- successfully solved all tasks given in the ESWC 2016 Top-k Shortest Path Challenge
- in future versions we intend to lazily build heaps and try to predict invalid paths earlier

Acknowledgement

This work was partially funded by the BMBF project Multimedia Opinion Mining (MOM: 01WI15002).

Contact:
Msc. Sven Hertling
Researcher - Knowledge Management Department
German Research Center for Artificial Intelligence
DFKI GmbH

Phone: +49 631 20575-5230
Mobile: +49 175 160 12 15
Mail: sven.hertling@dfki.de
Website: http://km.dfki.de/sven-hertling-msc