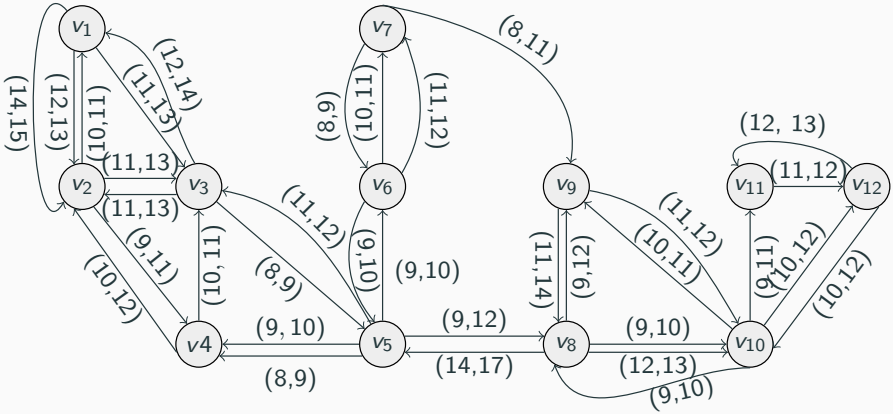
An Efficient Index for Reachability Queries in Public Transport Networks

Bezaye Tesfaye Nikolaus Augsten Mateusz Pawlik
Michael Böhlen Christian S. Jensen

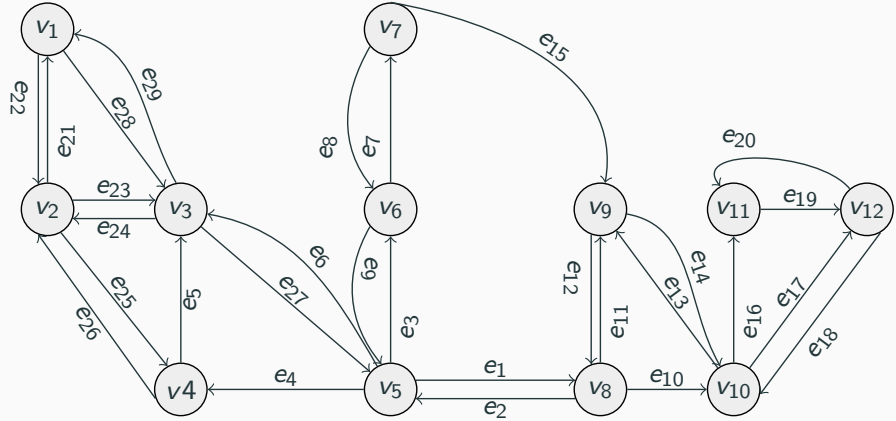
ADBIS: Advances in Databases and Information Systems, August 2020

Motivation



Path costs between stations depend on the start time

Motivation

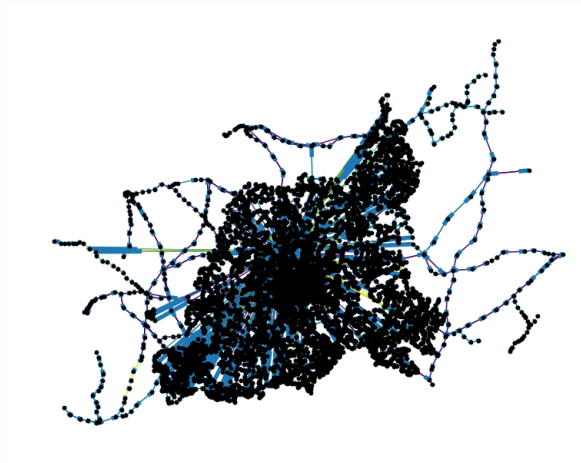


Path costs between nodes depend on the start time

Reachability Query

Reachability query in a temporal graph G**:**

- Which points of interst (POIs) are reachable given:
  - start time
  - start node
  - cost budget
- Application areas:
  - geomarketing
  - recommender systems
  - logistics, ...



Existing Approaches

1. **No-index**: expand edge by edge to find all reachable POIs in the graph
   - does not scale to large networks
2. **Shortest-path index**: issue a shortest path query for each POI
   - expensive index construction
   - does not scale to large numbers of POIs

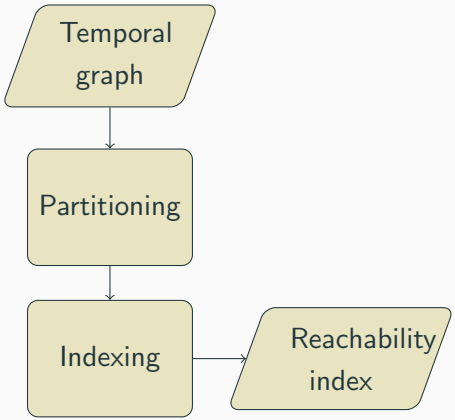
Goal: minimize the query time and the preprocessing time

Our Approach

- Preprocessing:
  - design a reachability index based on graph partitioning
- Query Processing:
  - apply cell by cell expansion using the reachability index

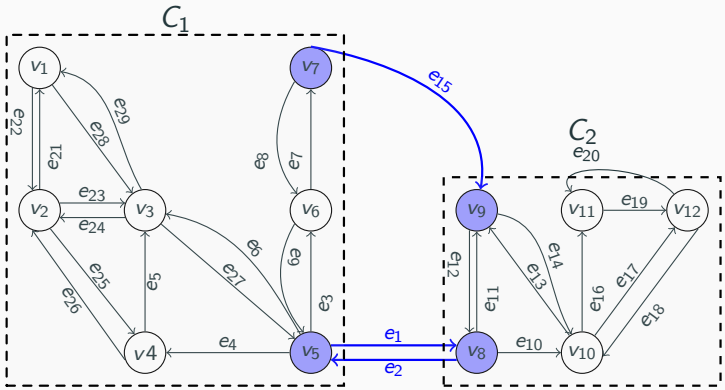
Preprocessing

- Building a reachability index
- Involves:
  - graph partitioning
  - construction of the index core
  - computing index cost function
  - inserting POIs



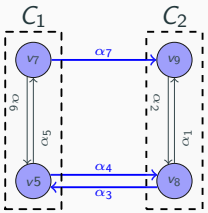
Graph Partitioning

- Given $G = (V, E, c)$, we partition V into a set of disjoint cells
  - **border edges** = edges that connect nodes from different cells
  - **border nodes** = end points of *border edges*

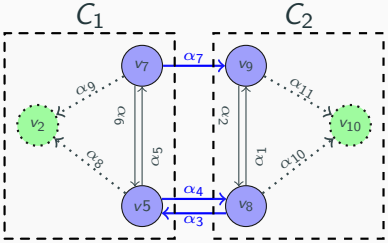


Reachability Index R

- An index core stores:
  - border nodes
  - border edges
  - an edge between each pair of border nodes within a cell
- Reachability index R = index core + POIs:
  - all nodes and edges in the index core
  - a set of POIs
  - an edge from each border node to each POI within a cell



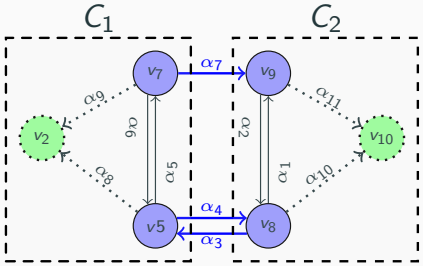
(a) Index core



(b) Reachability index R

Query Processing

- Cell by cell expansion (similar to Dijkstra's algorithm)
- Expand in index R from a query node at a specific start time within a given cost budget



- $RQ(R, v_5, 8, 6) = \{v_2, v_{10}\}$
  - $sp(v_5, v_2, t) = 4$ and $sp(v_5, v_{10}, t) = 5$

Experiments

- GTFS datasets and Synthetic

Dataset	#Nodes	#Edges	#Conn	#Part	#B-nodes		Part. size			#POIs	
					sum	avg	avg	min	max	sum	avg
Zurich	2,508	5,630	555,713	45	315	7.0	55	2	157	99	2.20
Berlin	12,984	34,791	1,348,070	50	1,241	24.8	259	2	921	567	11.34
Synthetic	145,188	433,272	31,042,468	44	1,245	28.3	3,299	831	4,037	7,176	163.00

Comparision with respect to index size and number of expanded edges:

- No-index (NI): expand with temporal Dijkstra in original graph
- Shortest-path (SP): precompute shortest path to all POIs (from each node)
- Reachability query (RQ): our solution

Evaluation

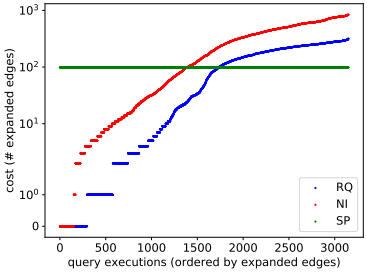
Reachability query = RQ, Shortest-path = SP, No-index = NI

Dataset	Algorithm	#Nodes	#Edges	#Connections
Zurich	*RQ*	414	4,021	421,268
	SP	2,508	248,292	55,015,587
	NI	2,508	5,630	555,713
Berlin	*RQ*	1,808	53,543	2,533,940
	SP	12,984	7,361,928	764,355,690
	NI	12,984	34,791	1,348,070
Synthetic	*RQ*	8,421	212,564	18,018,811
	SP	145,188	1,041,869,088	222,760,750,368
	NI	145,188	433,272	31,042,468

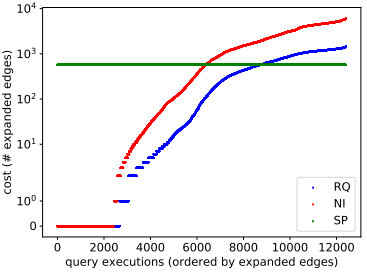
Index size (RQ) $<<$ Index size (SP)

Evaluation

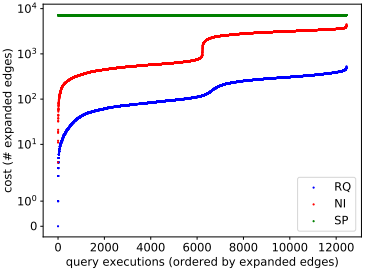
♣ **start times:** $\{8, 12, 16, 18, 22\}$**, cost budgets:** $\{60min, 120min\}$**, POIs: 5% of nodes**



(a) *Zurich*



(b) *Berlin*



(c) *Synthetic*

Summary

- Reachability index requires precomputation from and to border nodes
- Cell by cell expansion scales to large networks
- Network structure may affect the effectiveness of reachability index

Thank you!