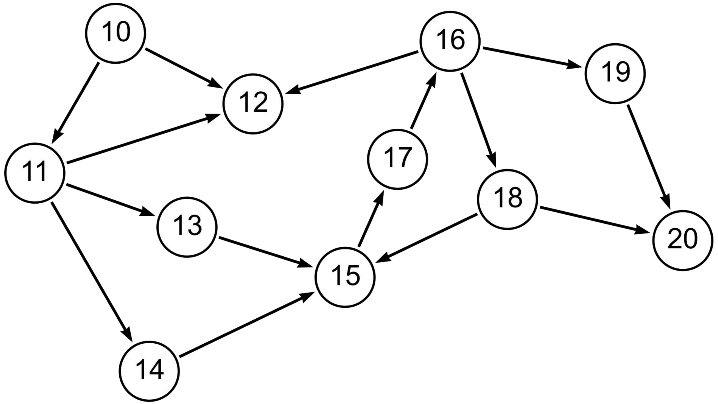
Reachability Indexes

Chao Zhang, Lyon 1 University

Graphs

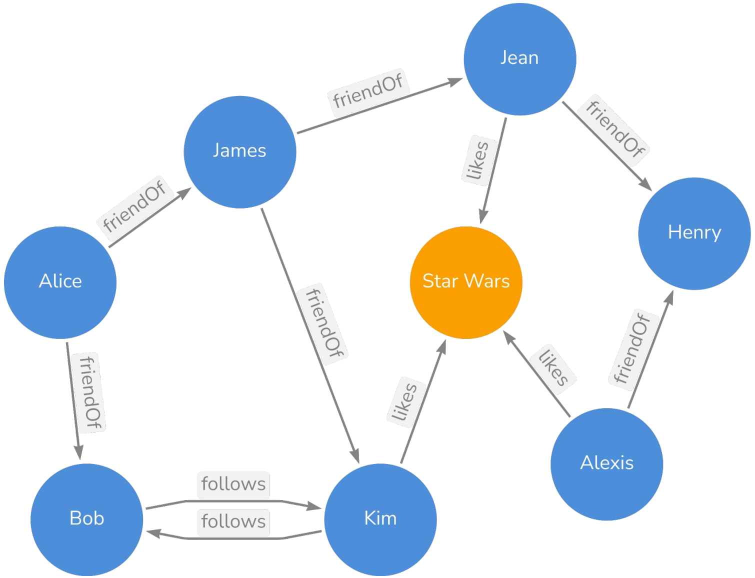
Plain reachability

Plain graph



Path-constraint reachability

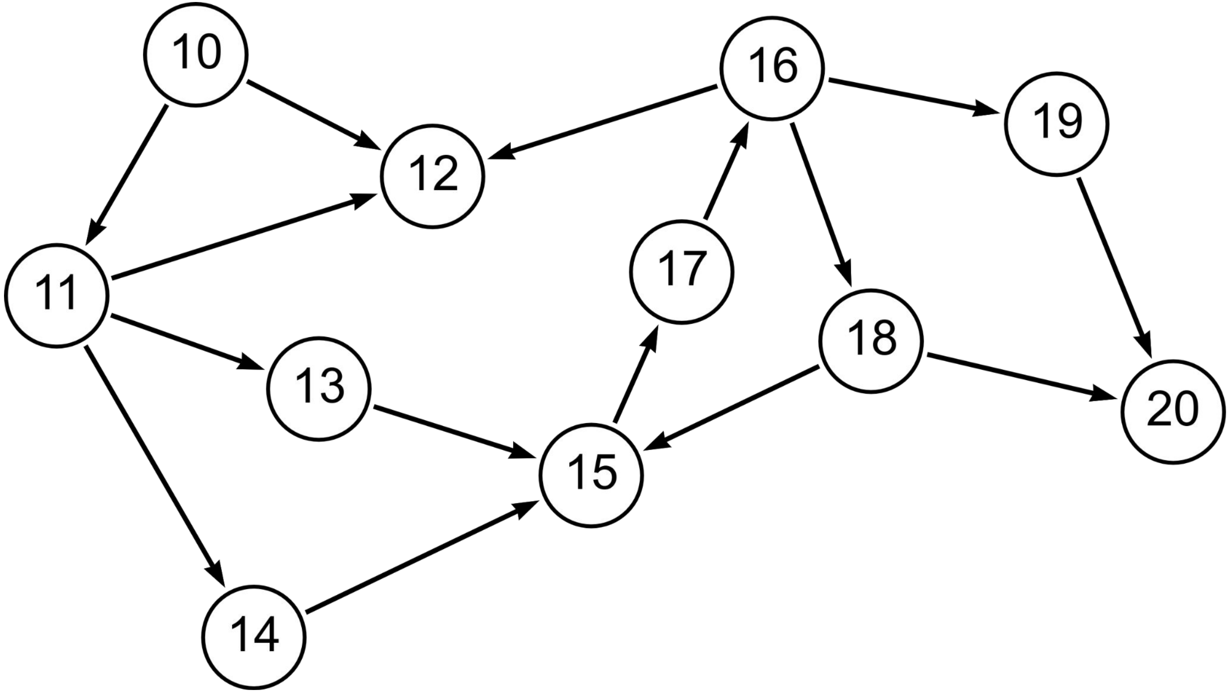
Edge-labeled graph



Agenda

1. Reachability on **plain** graphs
    a. A panoramic view of reachability indexes
    b. Milestones
2. Reachability on **edge-labeled** graphs
    a. Techniques
        - Alternation-based path constraints
        - Concatenation-based path constraints
    b. Challenges

Section I: Plain Reachability



Is there a path from vertex 14 to vertex 20?

Plain reachability query

- Q(s, t) on a directed graph G
  - Checking the existence of a path from s to t in G
- Boolean query
  - Either *true* or *false*
- Fundamental graph operator [Sah17]
  - Inferring the relationships among objects

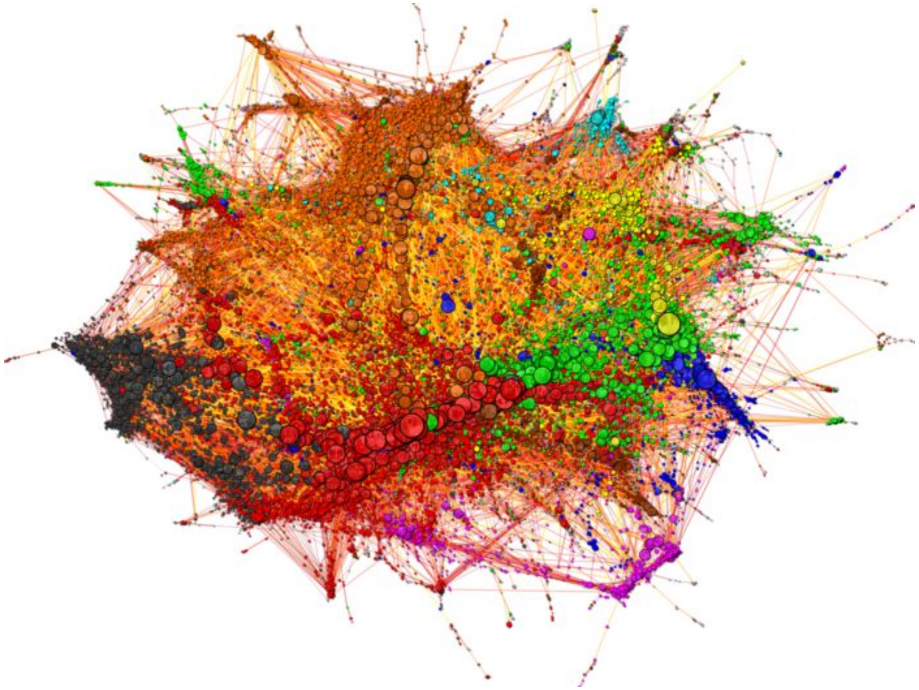
    E.g., querying protein-protein interaction in biology networks [Yil]

    E.g., querying related works in citation networks [Yil]

[Sah17] S. Sahu et al. The ubiquity of large graphs and surprising challenges of graph processing. VLDB J. 29(2-3): 595-618 (2020)
[Yil10] H. Yildirim et al. GRAIL: Scalable Reachability Index for Large Graphs. Proc. VLDB Endow. 3(1): 276-284 (2010)

Reachability query processing

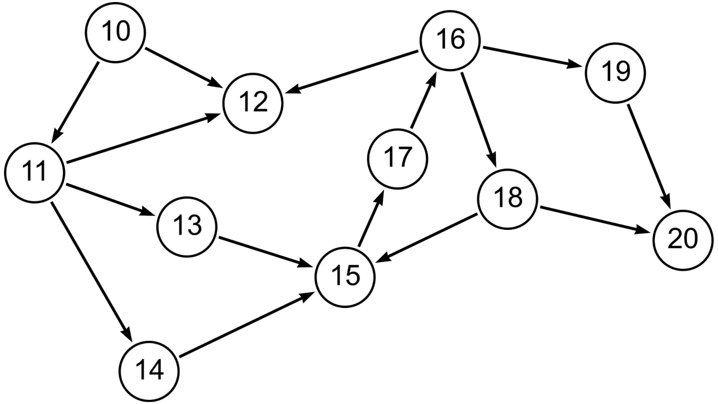
- Query evaluation
  - Online traversal: BFS, DFS, and BiBFS
  - Problem: graphs are large
- An index for reachability queries
  - Reachability index



Music recommendation graph: http://sixdegrees.hu/last.fm/interactive_map.html

Naive index: transitive closure

Example: Vertex 10 reaches vertex 20 as the cell is not empty



target

source

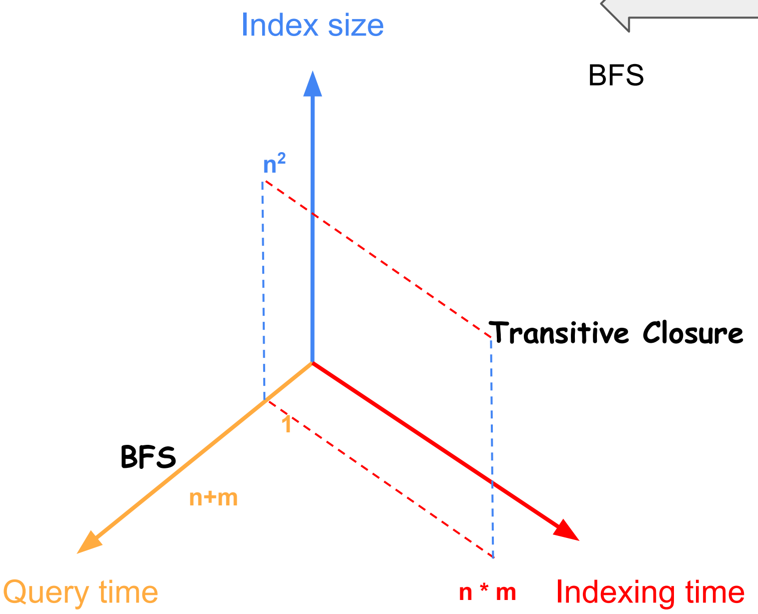
	10	11	12	13	14	15	16	17	18	19	20
10		T	T	T	T	T	T	T	T	T	T
11			T	T	T	T	T	T	T	T	T
12											
13			T			T	T	T	T	T	T
14			T			T	T	T	T	T	T
15			T			T	T	T	T	T	T
16			T			T	T	T	T	T	T
17			T			T	T	T	T	T	T
18			T			T	T	T	T	T	T
19											T
20											

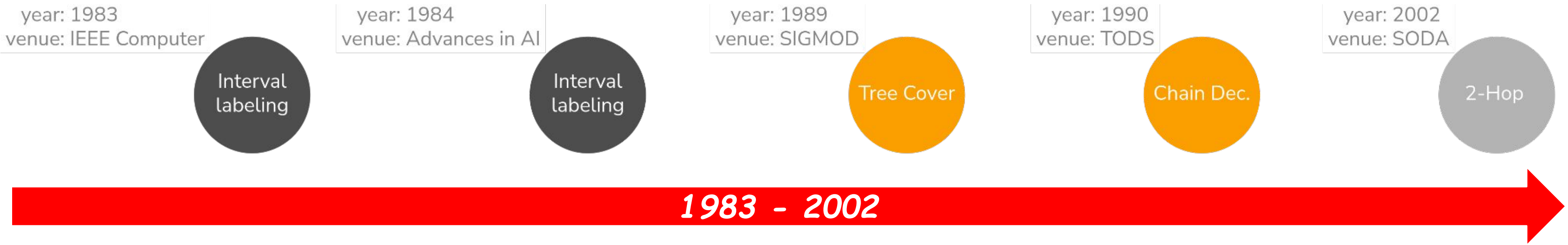
Complexity

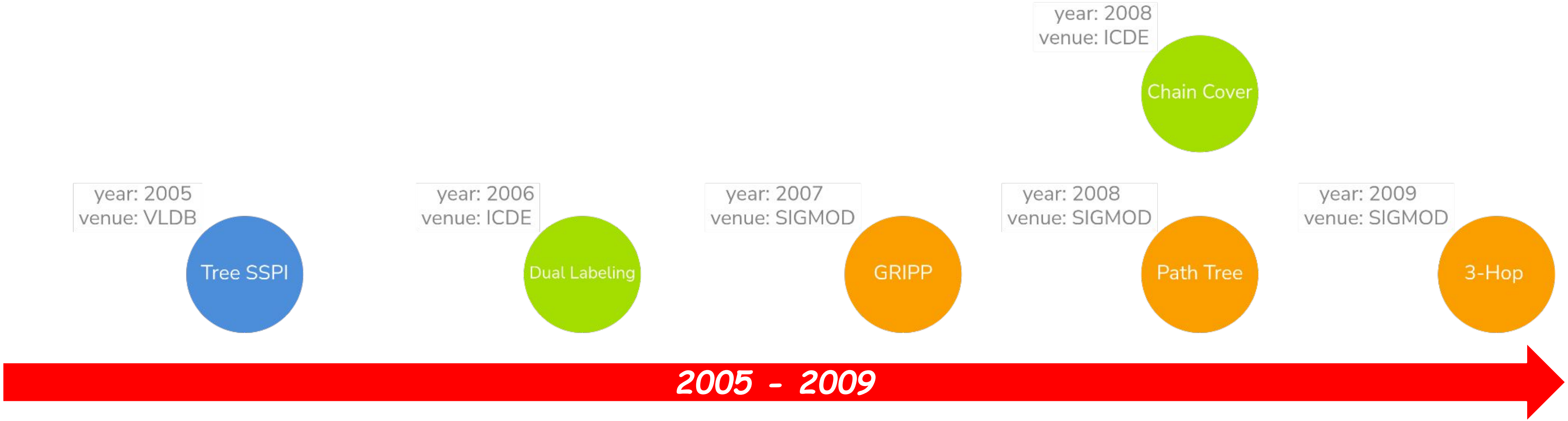
G (V, E)

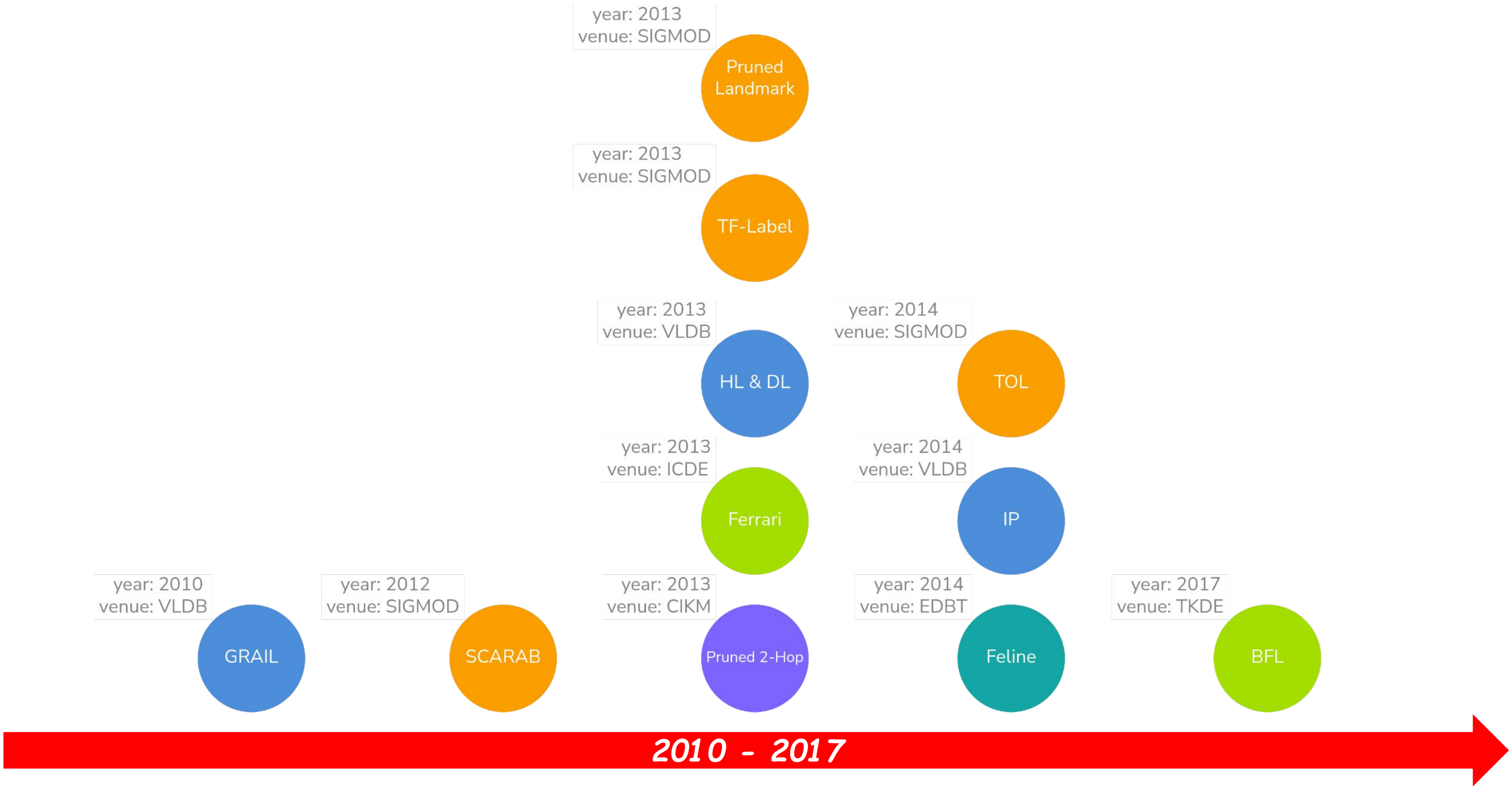
n = |V|

m = |E|




year: 1983
venue: IEEE Computer
Interval labeling
year: 1984
venue: Advances in AI
Interval labeling
year: 1989
venue: SIGMOD
Tree Cover
year: 1990
venue: TODS
Chain Dec.
year: 2002
venue: SODA
2-Hop
1983 - 2002


year: 2008
venue: ICDE
Chain Cover
year: 2005
venue: VLDB
Tree SSPI
year: 2006
venue: ICDE
Dual Labeling
year: 2007
venue: SIGMOD
GRIPP
year: 2008
venue: SIGMOD
Path Tree
year: 2009
venue: SIGMOD
3-Hop
2005 - 2009


year: 2013
venue: SIGMOD
Pruned Landmark
year: 2013
venue: SIGMOD
TF-Label
year: 2013
venue: VLDB
HL & DL
year: 2014
venue: SIGMOD
TOL
year: 2013
venue: ICDE
Ferrari
year: 2014
venue: VLDB
IP
year: 2010
venue: VLDB
GRAIL
year: 2012
venue: SIGMOD
SCARAB
year: 2013
venue: CIKM
Pruned 2-Hop
year: 2014
venue: EDBT
Feline
year: 2017
venue: TKDE
BFL
2010 - 2017


Tree Cover
2-Hop Labeling
Approximate TC
GO

Step 2

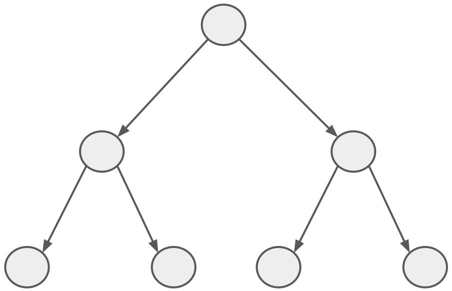
Interval Labeling

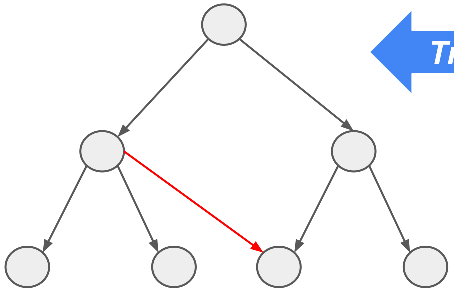
Step 3

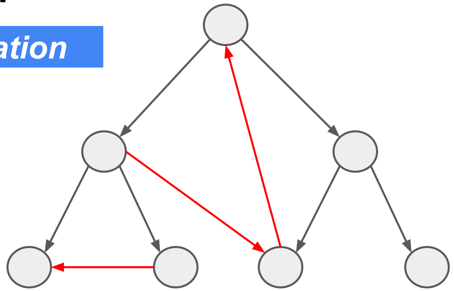
Tree Cover

Step 1

Transformation



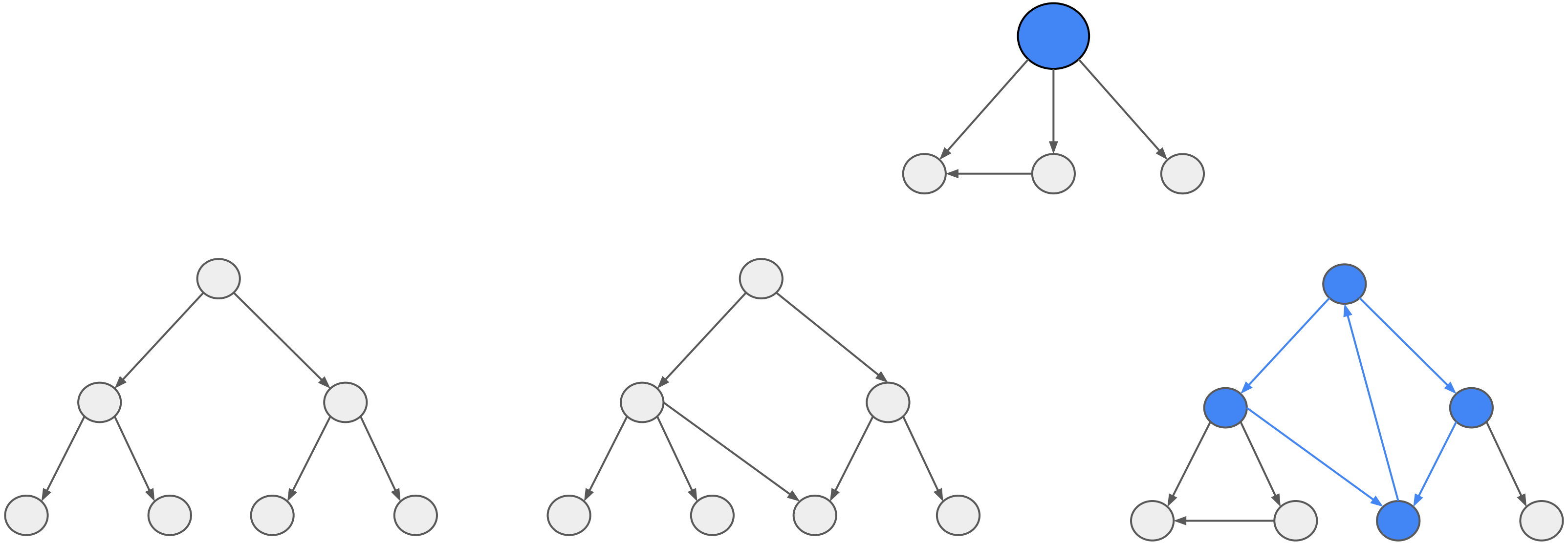




Tree

DAG

Cyclic Graph



Tree

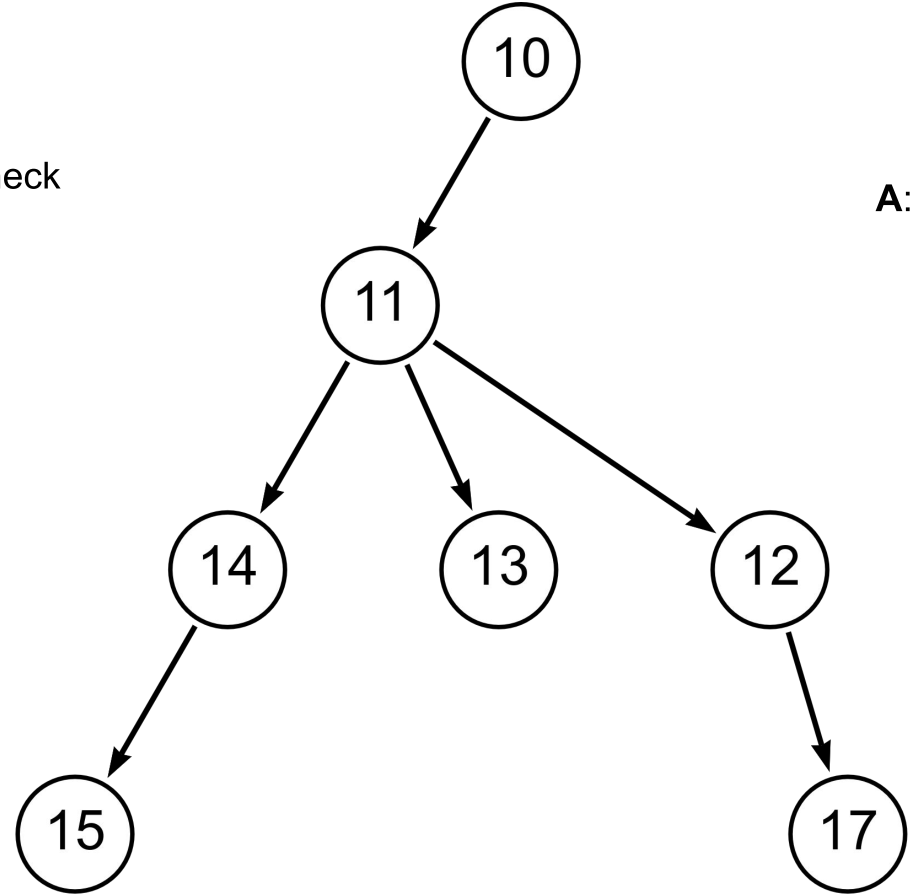
DAG

Tarjan's Algorithm [Tar72] to compute Strongly Connected Components

Cyclic Graph

[Tar72] R. Tarjan. Depth-First Search and Linear Graph Algorithms. SIAM J. Comput. 1(2): 146-160 (1972)

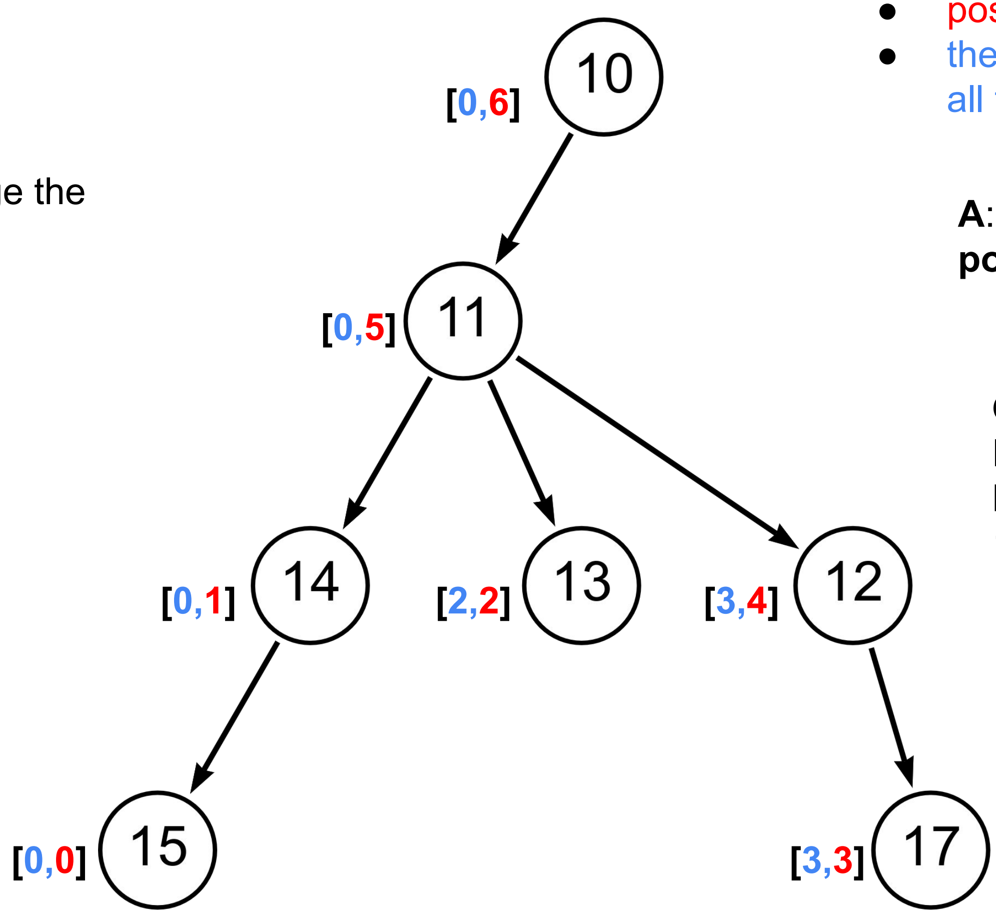
Q: How to efficiently check Q(10,14) on the tree?



A: Subtree containment

Interval labeling, for each v
- postorder number of v
- the smallest postorder number of all the successors of v

Q: How to leverage the property?



A: Interval schema based on **postorder traversal**

Q(10, 14) = True
Interval of 10: [0,6]
Postorder number of 14: 1
$1 \in [0,6]$

[Agr89] R. Agrawal et al. Efficient management of transitive relationships in large data and knowledge bases. SIGMOD Conference 1989: 253-262

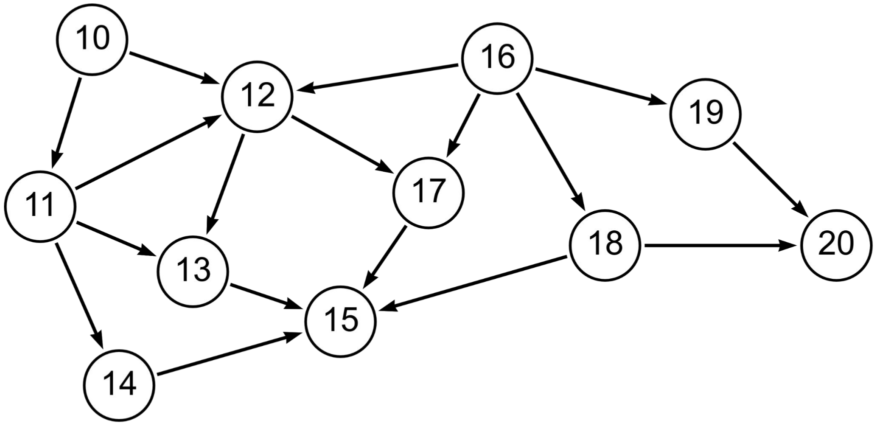
From Tree to DAG

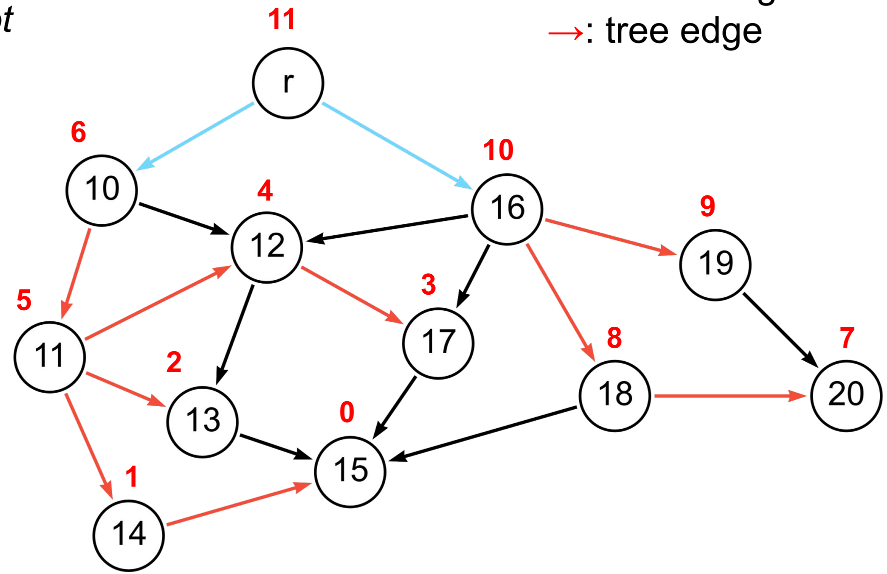
Q1: How about multiple trees?

Assigning a virtual root

r: virtual root
→: virtual edge
→: tree edge

DAG



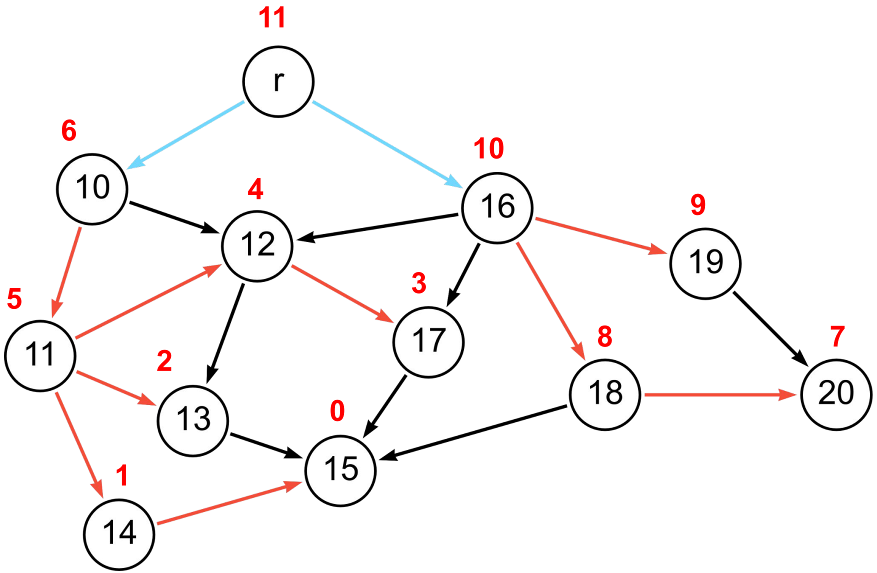


[Agr89] R. Agrawal et al. Efficient management of transitive relationships in large data and knowledge bases. SIGMOD Conference 1989: 253-262

From Tree to DAG

Q2: How about non-tree edges?

Inheriting the intervals



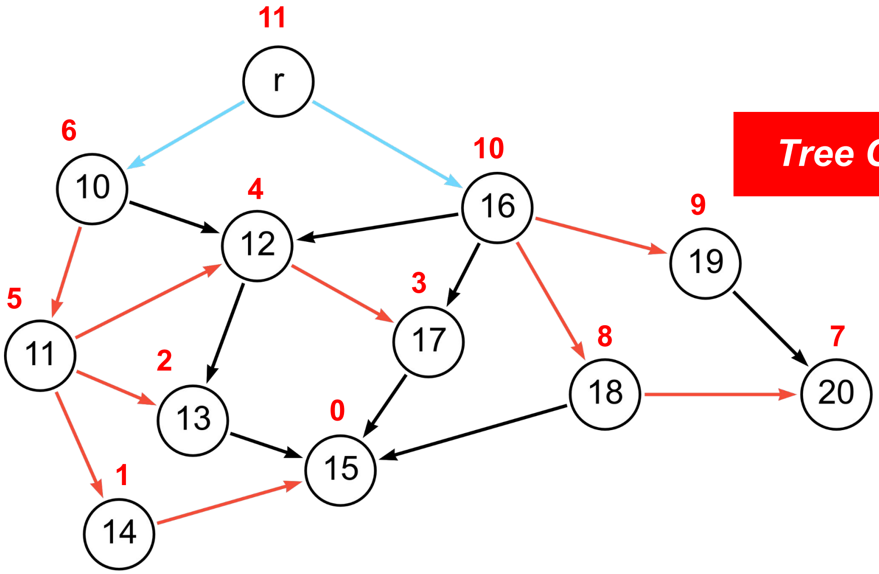
V	Interval
10	[0, 6]
11	[0, 5]
12	[3, 4]
13	[2, 2]
14	[0, 1]
15	[0, 0]
16	[7, 10]
17	[3, 3]
18	[7, 8]
19	[9, 9]
20	[7, 7]

[Agr89] R. Agrawal et al. Efficient management of transitive relationships in large data and knowledge bases. SIGMOD Conference 1989: 253-262

From Tree to DAG

Q2: How about non-tree edges?

Inheriting the intervals



Tree Cover

V	Interval
10	[0, 6]
11	[0, 5]
12	[3, 4], [2,2], [0,0]
13	[2, 2], [0, 0]
14	[0, 1]
15	[0, 0]
16	[7, 10], [3, 3], [0,0], [3, 4], [2,2]
17	[3, 3], [0,0]
18	[7, 8], [0, 0]
19	[9, 9], [7, 7]
20	[7, 7]

Merging intervals

[2, 4], [0, 0]

[7, 10], [2, 4], [0, 0]

[Agr89] R. Agrawal et al. Efficient management of transitive relationships in large data and knowledge bases. SIGMOD Conference 1989: 253-262

Complexity

- Index size: $O(n^2)$
- Indexing time: $O(nm)$
- Query time: $O(\log n)$
- Bottleneck:
  - A larger number of intervals caused by **non-tree edges**
- Q: how to reduce the number of intervals?

Reducing the number of intervals

- Bounding the number of intervals
  - GRAIL [Yil10]: exactly k intervals by computing k spanning trees
  - Ferrari [Seu13]: at most k intervals by merging non-adjacent intervals
- Incomplete indexes
  - **False positives** for query processing using indexes
- Resort to online search
  - Guided DFS by querying the incomplete indexes

[Yil10] H. Yildirim et al. GRAIL: Scalable Reachability Index for Large Graphs. Proc. VLDB Endow. 3(1): 276-284 (2010)
[Seu13] S. Seufert et al. FERRARI: Flexible and efficient reachability range assignment for graph indexing. ICDE 2013: 1009-1020

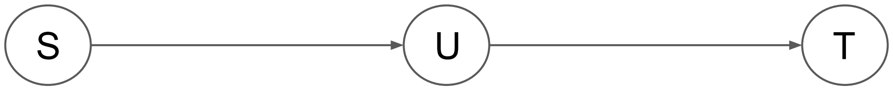
Other techniques based on tree cover

- Dual-labeling [Wan06]
  - Compressing transitive closure for non-tree edges
- GRIPP [Tri07]
  - Recursive querying intervals of rooted spanning trees

[Wan06] H. Wang et al. Dual Labeling: Answering Graph Reachability Queries in Constant Time. ICDE 2006: 75
[Tri07] S. Tril et al. Fast and practical indexing and querying of very large graphs. SIGMOD Conference 2007: 845-856


Tree Cover
2-Hop Labeling
Approximate TC
GO

Rethinking of transitive closure



target

source

	S	U	T
S		1	1
U			1
T			

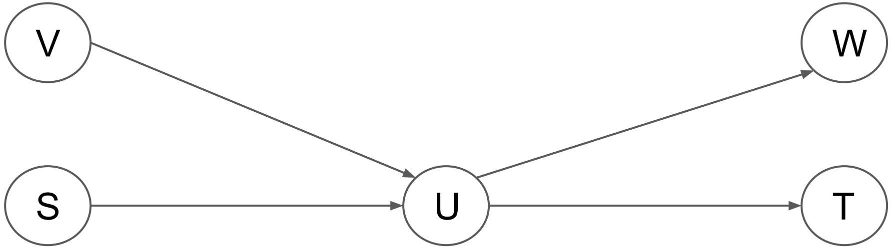
We can derive the existence of p(s, t) using p(s, u) and p(u, t)

Rethinking of transitive closure

target

source

	S	U	T	V	W
S		1	1		1
U			1		1
T					
V		1	1		1
W					



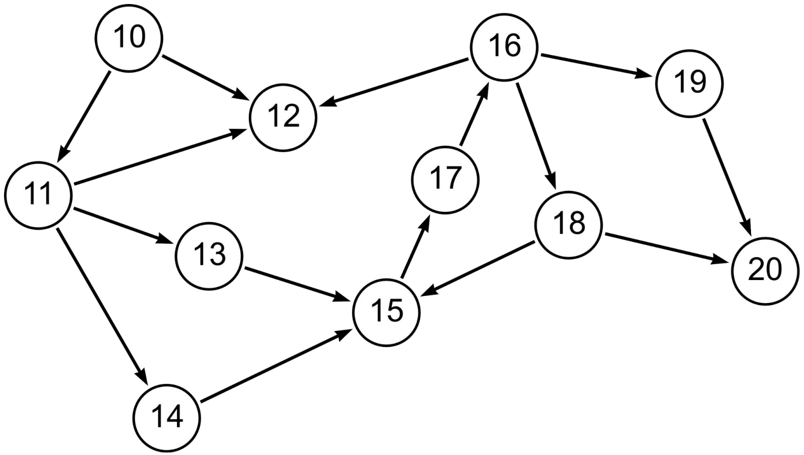
With the deriving, we only need to record p(s, u), p(v, u), p(u, w), and p(u, t).

2-Hop labeling

- Assigning $L(v) = (L_{in}(v), L_{out}(v))$ for each v in G,
  - $\forall\ u \in \mathbf{L_{in}(v)}$, $\exists$ a path **from u to v**
  - $\forall\ w \in \mathbf{L_{out}(v)}$, $\exists$ a path **from v to w**
- Vertex s reaches t in G, if and only if
  - Case 1: $\exists\ t \in L_{out}(s)$, or
  - Case 2: $\exists\ s \in L_{in}(t)$, or
  - **Case 3**: $L_{in}(t) \cap L_{out}(s) \neq \emptyset$
- Index size: $\sum_{v \in V} |L_{in}(v)| + |L_{out}(v)|$

[Coh02] E. Cohen et al. Reachability and distance queries via 2-hop labels. SODA 2002: 937-946

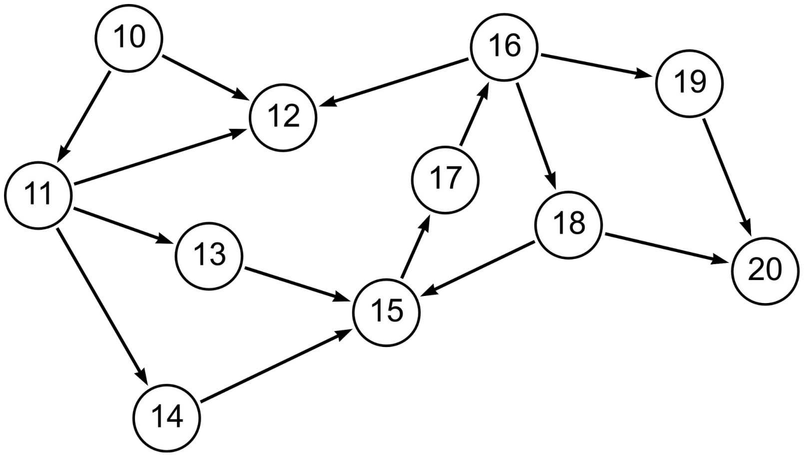
2-hop labeling



v	$L_{in}(v)$	$L_{out}(v)$
10	∅	11, 12, 15
11	∅	15
12	10, 11, 15, 16	∅
13	11	15
14	11	15
15	∅	15
16	17, 15	12, 15, 18
17	15	18
18	15	15
19	15	∅
20	15, 17, 18	∅

[Coh02] E. Cohen et al. Reachability and distance queries via 2-hop labels. SODA 2002: 937-946

2-hop labeling



v	$L_{in}(v)$	$L_{out}(v)$
10	∅	11, 12, 15
11	∅	15
12	10, 11, 15, 16	∅
13	11	15
14	11	15
15	∅	15
16	17, 15	12, 15, 18
17	15	18
18	15	15
19	15	∅
20	15, 17, 18	∅

Q(10, 20) = true, $L_{out}(10) \cap L_{in}(20) = 15$

Q(15, 18) = true, $15 \in L_{in}(18)$

Q(16, 13) = false

[Coh02] E. Cohen et al. Reachability and distance queries via 2-hop labels. SODA 2002: 937-946

Minimum 2-hop labeling

- The number of Case 3 should be **maximized**
- **Minimum** 2-hop: the one with the minimum index size
- NP-hard problem [1]
- Approximated algorithm [1]
    - Bounded by a logarithmic factor
    - Complexity
        - **Indexing time**: $O(n^4)$
        - Index size: $O(nm^{1/2})$
        - Query time: $O(m^{1/2})$

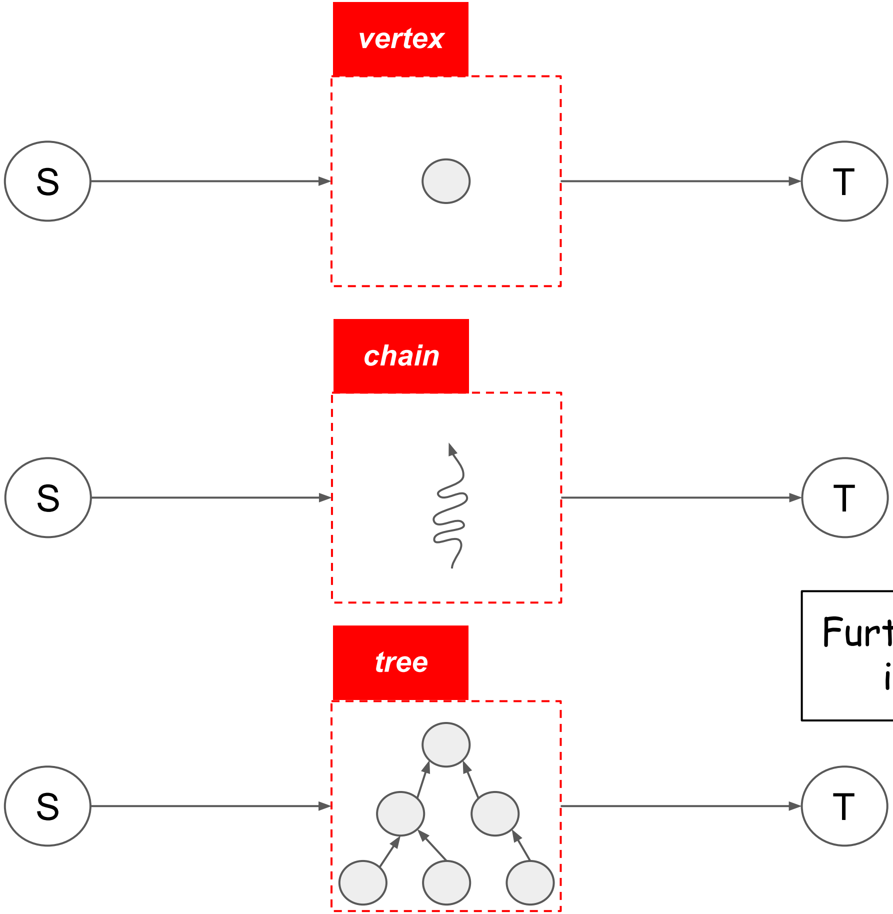
Impractical for
real-world large graphs

[Coh02] E. Cohen et al. Reachability and distance queries via 2-hop labels. SODA 2002: 937-946

Advanced 2-hop indexing heuristics

- TFL [Che13]
  - Recursive topological folding over DAG
- DL [Jin13]
  - Vertex order for non-redundant hop vertices
- PLL [Aki13]
  - Greedy indexing according to vertex degree
- TOL [Zhu14]
  - General total order for indexing

[Che13] J. Cheng et al. TF-Label: a topological-folding labeling scheme for reachability querying in a large graph. SIGMOD Conference 2013: 193-204
[Jin13] R. Jin et al. Simple, Fast, and Scalable Reachability Oracle. Proc. VLDB Endow. 6(14): 1978-1989 (2013)
[Aki13] E. Akiba et al. Fast exact shortest-path distance queries on large networks by pruned landmark labeling. SIGMOD Conference 2013: 349-360
[Zhu14] A. Zhu et al. Reachability queries on large dynamic graphs: a total order approach. SIGMOD Conference 2014: 1323-1334



2-Hop

3-Hop [Jin09]

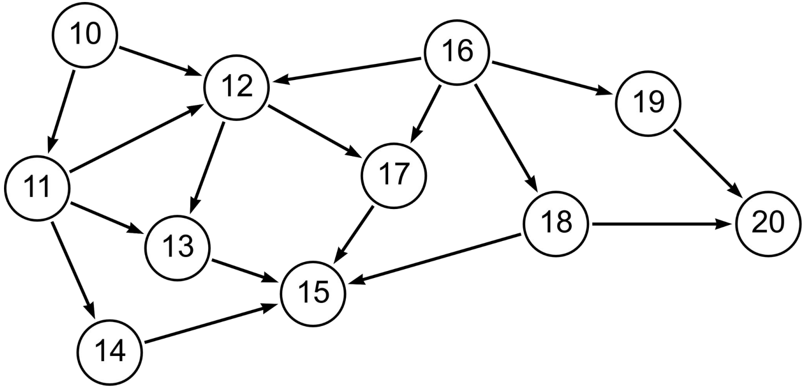
Further reducing index size

Path-Hop [Cai10]

[Jin09] R. Jin et al. 3-HOP: a high-compression indexing scheme for reachability query. SIGMOD Conference 2009: 813-826
[Cai10] J. Cai et al. Path-hop: efficiently indexing large graphs for reachability queries. CIKM 2010: 119-128


Tree Cover
2-Hop Labeling
Approximate TC
GO

Rethinking of transitive closure



- out(v): v and all the vertices that v can reach
- **If** u reaches v, **then** out(v) $\subseteq$ out(u)
- Example: 10 reaches 13,
  - out(10) = {10, 11, 12, **13**, 14, **15**, 17}
  - out(13) = {**13**, **15**}
- **If** out(v) $\nsubseteq$ out(u), **then** u does not reach v
- Similarly, **if** in(u) $\nsubseteq$ in(v), **then** u does not reaches v, where in(v) denotes v and all the vertices that can reach v

How to leverage the contrapositive conditions?

Membership testing

K-min-wise independent permutation

Reachability Querying: An Independent Permutation Labeling Approach

VLDB'14

Hao Wei, Jeffrey Xu Yu, Can Lu
Chinese University of Hong Kong
Hong Kong, China

Ruoming Jin
Kent State University
Kent, OH, USA

Bloom filter

Reachability Querying: Can It Be Even Faster?

TKDE'17

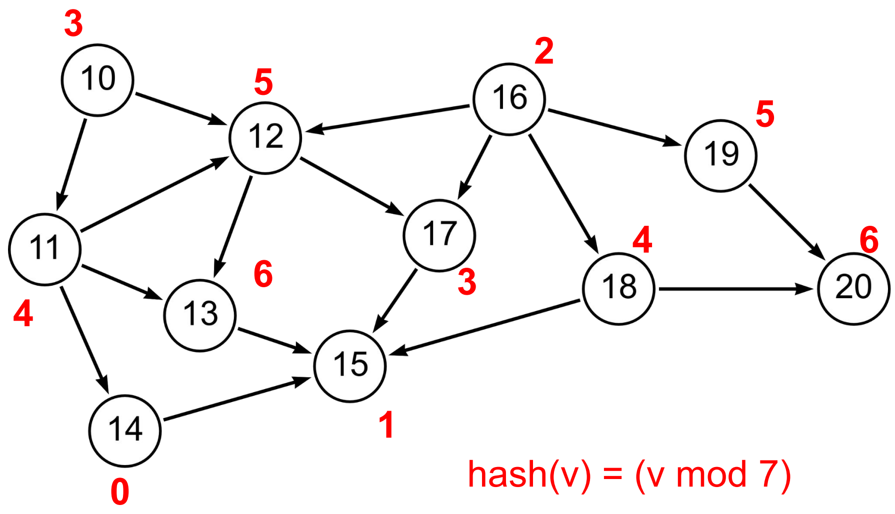
Jiao Su[†‡], Qing Zhu[†], Hao Wei[‡], and Jeffrey Xu Yu[‡]

[†] *Renmin University of China, Beijing, China;* [‡] *The Chinese University of Hong Kong, Hong Kong*

Bloom filter labeling

- Compute in(v) and out(v) for each v
  - In in(v) and out(v), recording the **hash codes** of vertices
- Query processing
  - set containment testing
- False positives require online traversal
  - Guided DFS with recursively querying the index

Bloom Filter Labeling



hash(v) = (v mod 7)

Q(12, 18): False because out(18) ⊄ out(12)

Q(11, 18): True, but false positive
Guided DFS leads to False

V	in(v)	out(v)
10	3	0,1,3,4,5,6
11	3,4	0,1,3,4,5,6
12	2,3,4,5	1,3,5,6
13	3,4,5,6	1,6
14	0,3,4	0,1
15	0,1,2,3,4,5,6	1
16	2	1,2,3,4,5,6
17	2,3,4,5	1,3
18	2,4	1,4,6
19	2,5	5,6
20	2,4,5	6

Other reachability techniques

- Path-tree labeling [Jin08]: *path partition + two-dimension labeling over a planar graph*
- SCRAB [Jin12]: *reachability backbone + reachability through backbone vertices*
- HL [Jin13]: *recursive reachability backbones*
- Feline [Vel14]: *dominance drawing (no false negatives) + online search*
- Preach [Mer14]: *contraction hierarchies + bidirectional online search*
- O’Reach [Han21]: *partial hop labeling + topological order + existing indexes*

[Jin08] R. Jin et al. Efficiently answering reachability queries on very large directed graphs. SIGMOD Conference 2008: 595-608
[Jin12] R. Jin et al. SCARAB: scaling reachability computation on large graphs. SIGMOD Conference 2012: 169-180
[Jin13] R. Jin et al. Simple, Fast, and Scalable Reachability Oracle. Proc. VLDB Endow. 6(14): 1978-1989 (2013)
[Vel14] R. Veloso et al. Reachability Queries in Very Large Graphs: A Fast Refined Online Search Approach. EDBT 2014: 511-522
[Mer14] F. Merz et al. PReaCH: A Fast Lightweight Reachability Index Using Pruning and Contraction Hierarchies. ESA 2014: 701-712
[Han21] K.Hanauer et al. O'Reach: Even Faster Reachability in Large Graphs. SEA 2021: 13:1-13:24

Readings

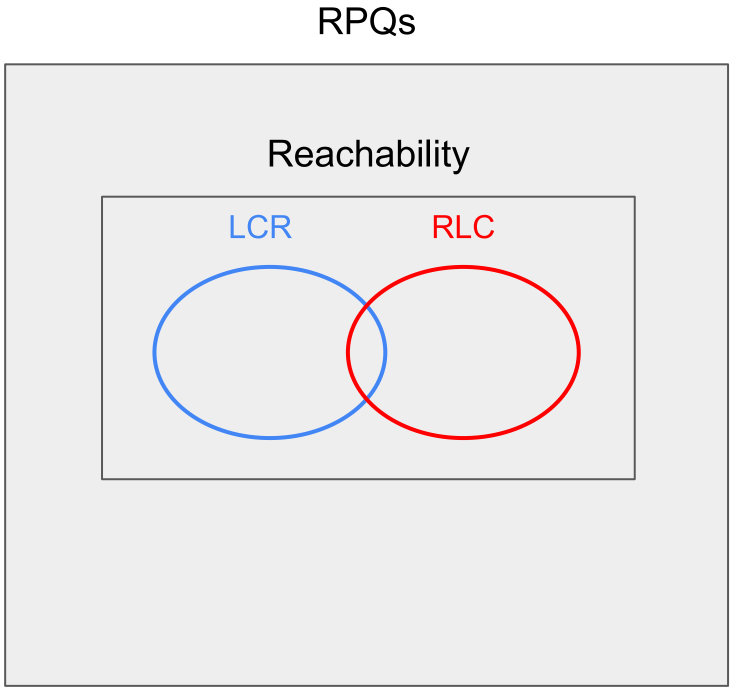
- 2 minutes
  - T. Özsu. *Graph Processing: A Panoramic View and Some open Problems*. Keynote at VLDB'19. (The section on reachability queries)
- 10 minutes
  - J. Su et al. *Reachability Querying: Can It Be even Faster?* In TKDE'17. (The related work section)
- Half a day
  - J. Xu yu et al. *Graph Reachability Queries: A Survey.* Managing and Mining Graph Data 2010.
- One day
  - A. Bonifati et al. *Querying Graphs*. Morgan & Claypool Publishers 2018. (Chapter 6.5: Reachability Indexing)
- Unlimited time
  - 9 SIGMOD/TODS + 4 VLDB + 4 ICDE/TKDE + 1 SODA + 1 EDBT, etc.

Section II: Path-Constraint Reachability

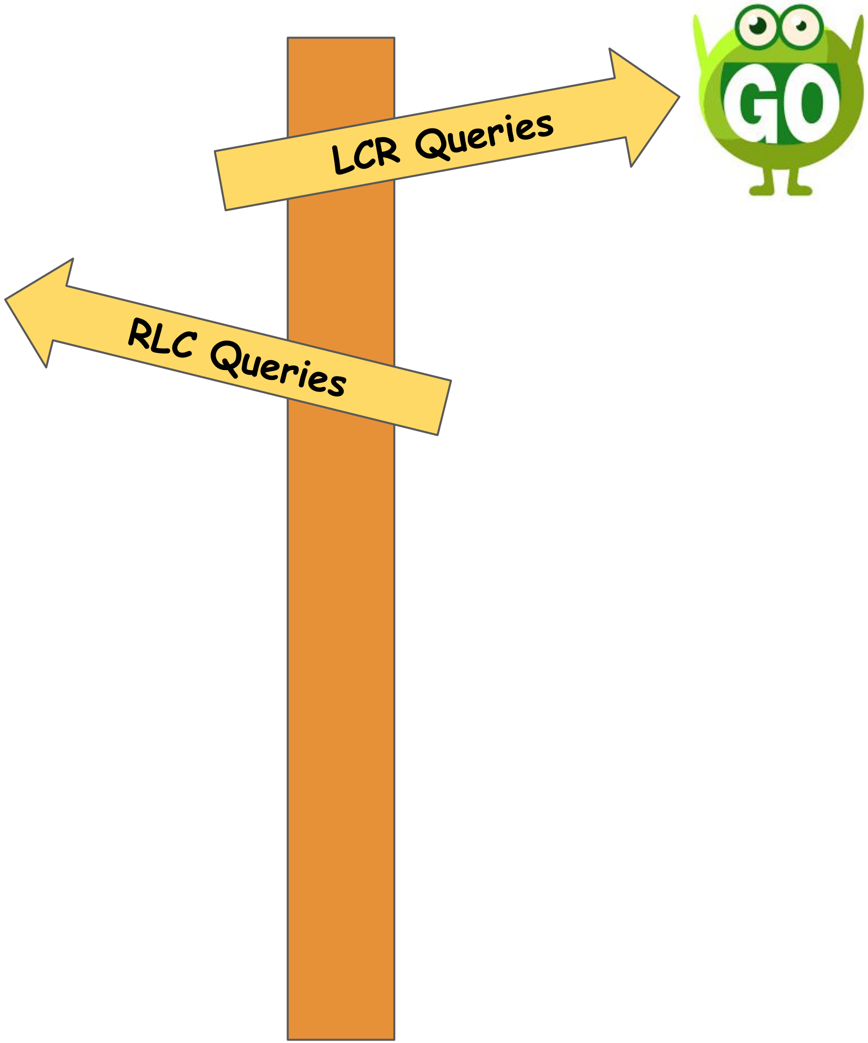
Reachability queries with path-constraints

The overlapping: a single label under the Kleene operator.

- Regular path queries (RPQs):
  - Having a regular expression as a constraint [Ang17]
- Reachability:
  - Checking the existence of a path that can satisfy a path constraint
- The Kleene operator: either * or +
- Two types (so far)
  - LCR: **alternation**-based reachability
  - RLC: **concatenation**-based reachability



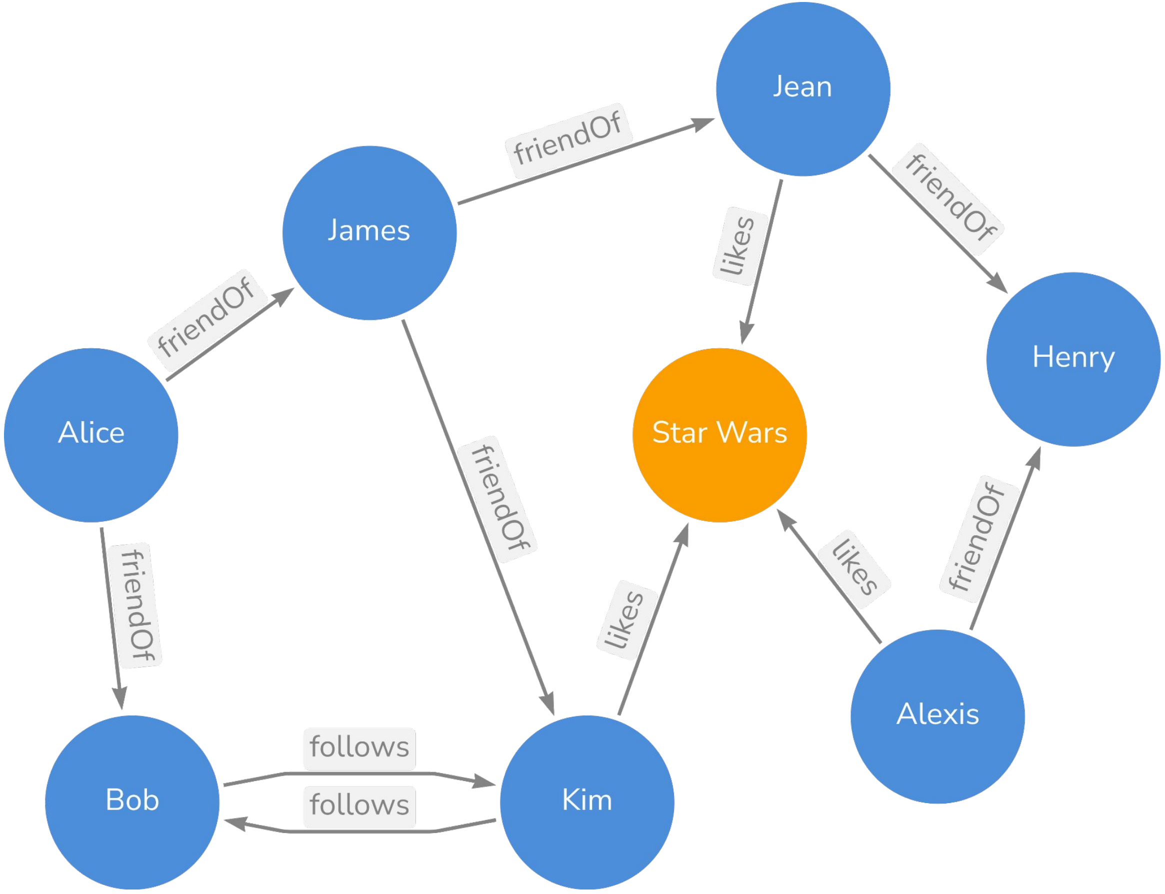
[Ang17] R. Angles et al. Foundations of Modern Query Languages for Graph Databases. ACM Comput. Surv. 50(5): 68:1-68:40 (2017)


LCR Queries
RLC Queries
GO

LCR (label-constrained reachability) queries

- LC (label constraint)
  - $(l_1 \cup \ldots \cup l_k)^+$, where $\cup$ is **disjunction**
- LCR query (s, t, LC)
  - Checking whether s reaches t
  - Checking whether **the path only contains edges with labels in the LC**
- Boolean query
  - Returning either True or False

- Supported languages
  - SPARQL
  - PGQL
  - openCypher

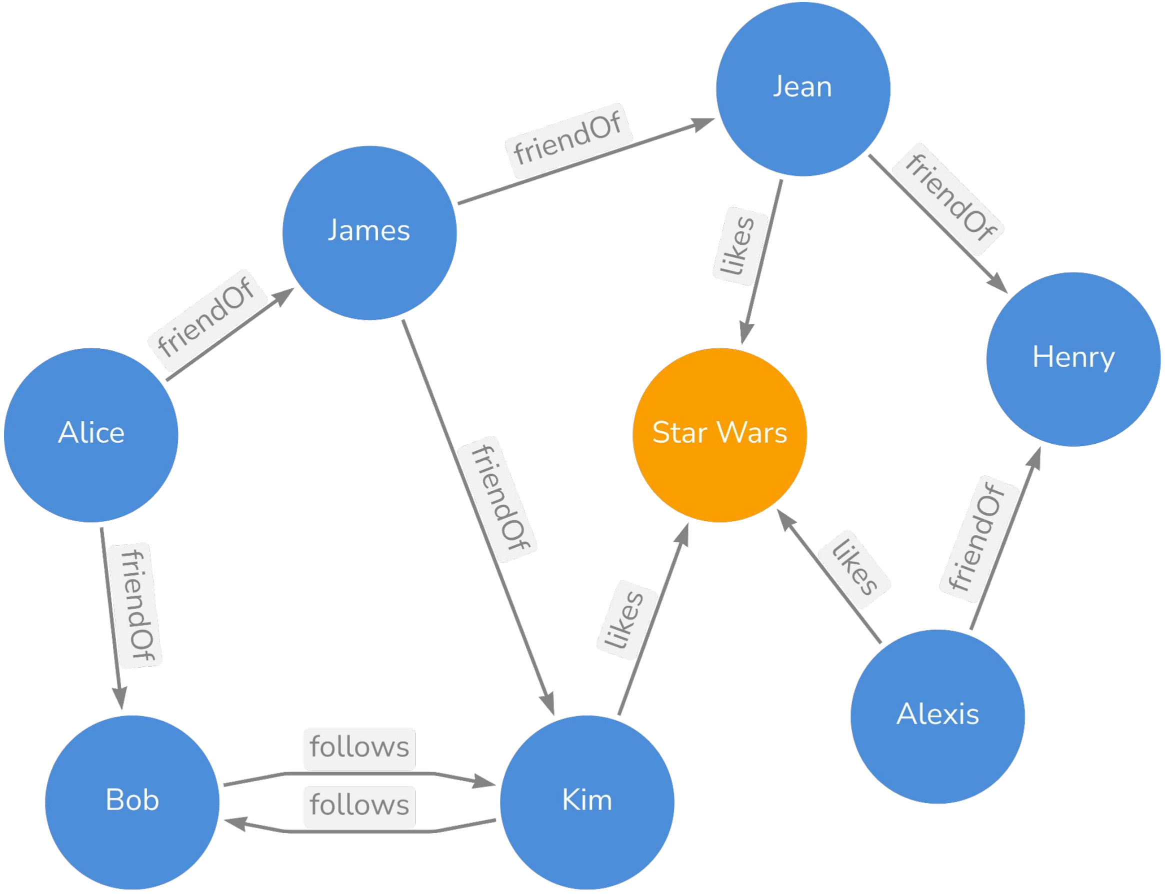


Does Alice reach Henry under the constraint {friendOf, follows}?

True

SPARQL

```
ASK
WHERE{
  :Alice (:friendOf|:follows)+ :Henry
}
```



Does Bob reach Star Wars
under the constraint
{friendOf, likes}?

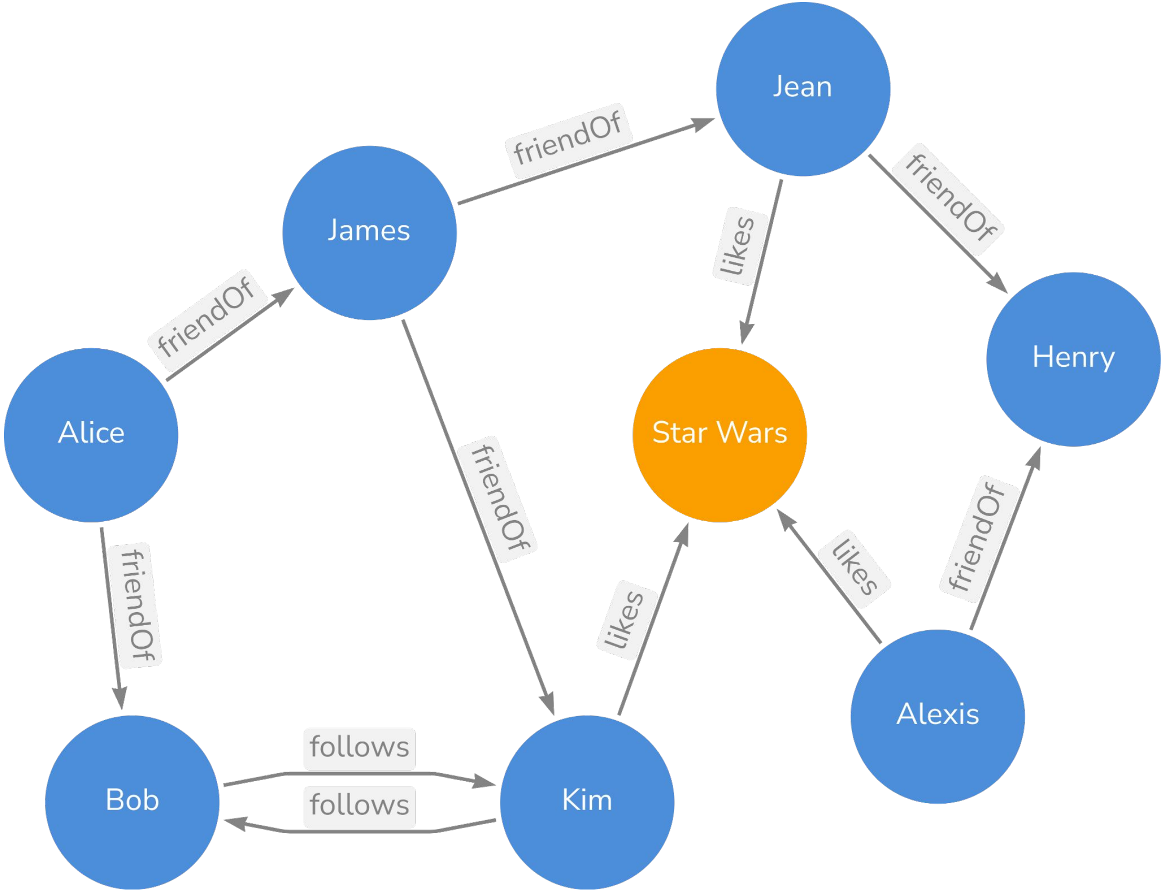
False

SPARQL

```
ASK
WHERE{
  :Bob (:friendOf|:likes)+ :Star_Wars
}
```

LCR query evaluation

- Online traversal
  - DFS, BFS, or BiBFS, visiting only edges with labels in the LC
  - Unfeasible for large graphs
- An index for LCR queries
  - LCR indexes
- Index-based evaluation for Q(s, t, LC)
  - **Path-label set** from s to t is mandatory
- Redundancy of path-label sets?



Two path-label sets from Alice to Kim

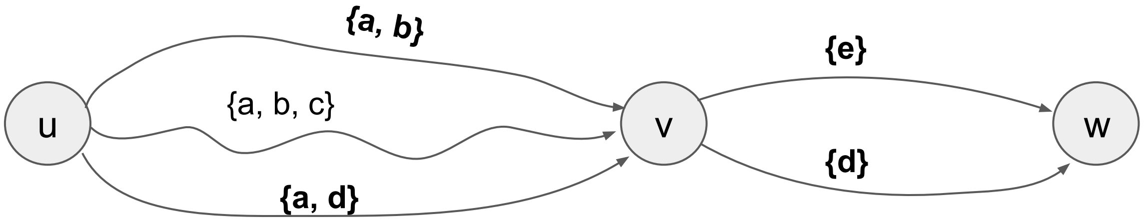
1. {friendOf}
2. {friendOf, follows}

Do we need to record both of them?

{friendOf} ⊂ {friendOf, follows} ⊆ a given constraint

Sufficient path-label set (SPLS)

- Definition [Jin10]
  - The **minimal subsets** of all the path-label sets from u to v



- **Free for merging** [1], i.e., *distributive*
  - Computing **SPLS**(p(u, w)) by using **SPLS**(p(u, v)) and **SPLS**(p(v, w))
  - SPLS from u to w: {a, b, e}, {a, b, d}, ~~{a, b, c, e}, {a, b, c, d}~~, {a, d, e}, and {a, d}

[Jin10] R. Jin et al. Computing label-constraint reachability in graph databases. SIGMOD Conference 2010: 123-134

GTC (Generalized transitive closure)

- GTC [1]: transitive closure with sufficient path-label set
  - For each (u, v):
    - recording whether u reaches v, and
    - SPLS(u, v)
- Problems:
  - Too much time to compute
  - Too much space to store
- How to efficiently compute and effectively compress GTC?

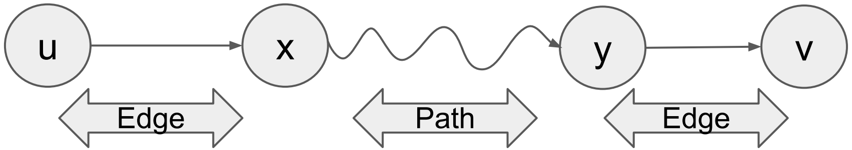
[Jin10] R. Jin et al. Computing label-constraint reachability in graph databases. SIGMOD Conference 2010: 123-134


year: 2010
venue: SIGMOD
Jin et al
year: 2014
venue: Information Systems
Zou et al
year: 2017
venue: SIGMOD
Valstar et al
year: 2020
venue: VLDB
Peng et al
year: 2021
venue: TODS
Chen et al
2010 - 2021

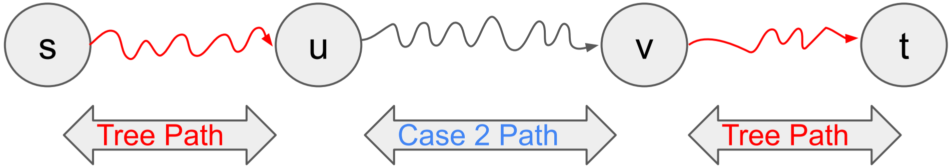
Jin et al

GTC compression using spanning tree

- Path characterization [1]:
  - Case 1: (u, x) or (y, v) is a tree edge
  - Case 2: neither (u, x) nor (y, v) is a tree edge
    - *Partial GTC*



- Query processing:
  - Case 2: partial GTC
  - Case 1: spanning tree + partial transitive closure

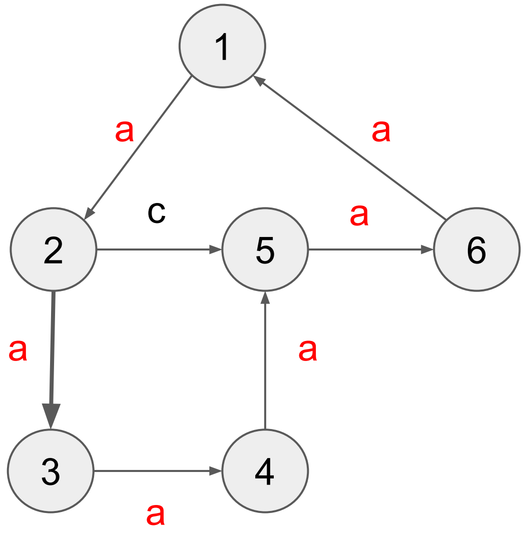


[Jin10] R. Jin et al. Computing label-constraint reachability in graph databases. SIGMOD Conference 2010: 123-134

Zou et al

Efficient GTC computation

- Observations:
  - redundant path-label sets do not need to be expanded
- Dijkstra-like algorithm [Zou14]
  - Simulating distance using **distinct** labels
- Example:
  - two path-label sets from 1 to 5 {a} and {a,c}
  - {a,c} can be pruned



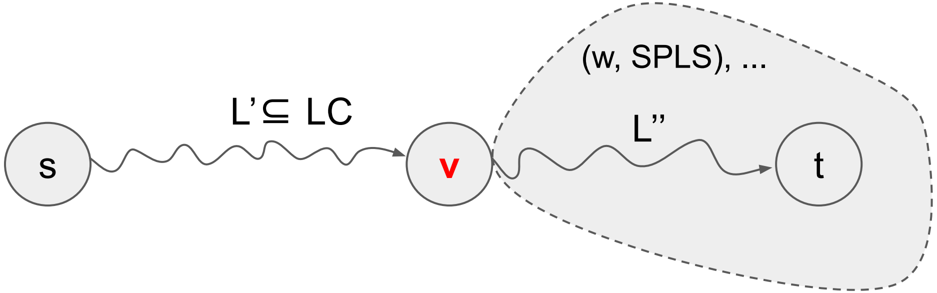
[Zou14] L. Zou et al. Efficient processing of label-constraint reachability queries in large graphs. Inf. Syst. 40: 47-66 (2014)

year: 2017
venue: SIGMOD

Valstar et al

Landmark index

- **Landmark** vertices
    - High degree vertices, e.g., hubs
- Landmark indexing [Val17]
    - Computing GTC for each landmark
- Query processing
    - BFS + Index lookup



Q(s, t, L): True, if L’’ ⊆ LC

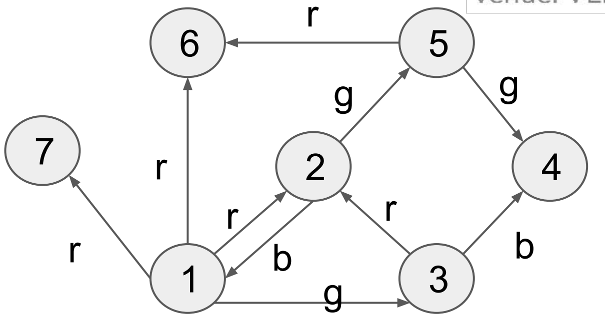
[Val17] L. Valstar et al. Landmark Indexing for Evaluation of Label-Constrained Reachability Queries. SIGMOD Conference 2017: 345-358

Label constrained 2-hop labeling

year: 2020
venue: VLDB

Peng et al

- The free for merging properties
  - SPLS
  - 2-hop labeling
- LC 2-hop [Pen20]
  - SPLS + PLL [Aki13]
- Example: Q(3, 6, {r, b})
  - (1, {r,b}) in $L_{out}(3)$
  - (1,{r}) in $L_{in}(6)$

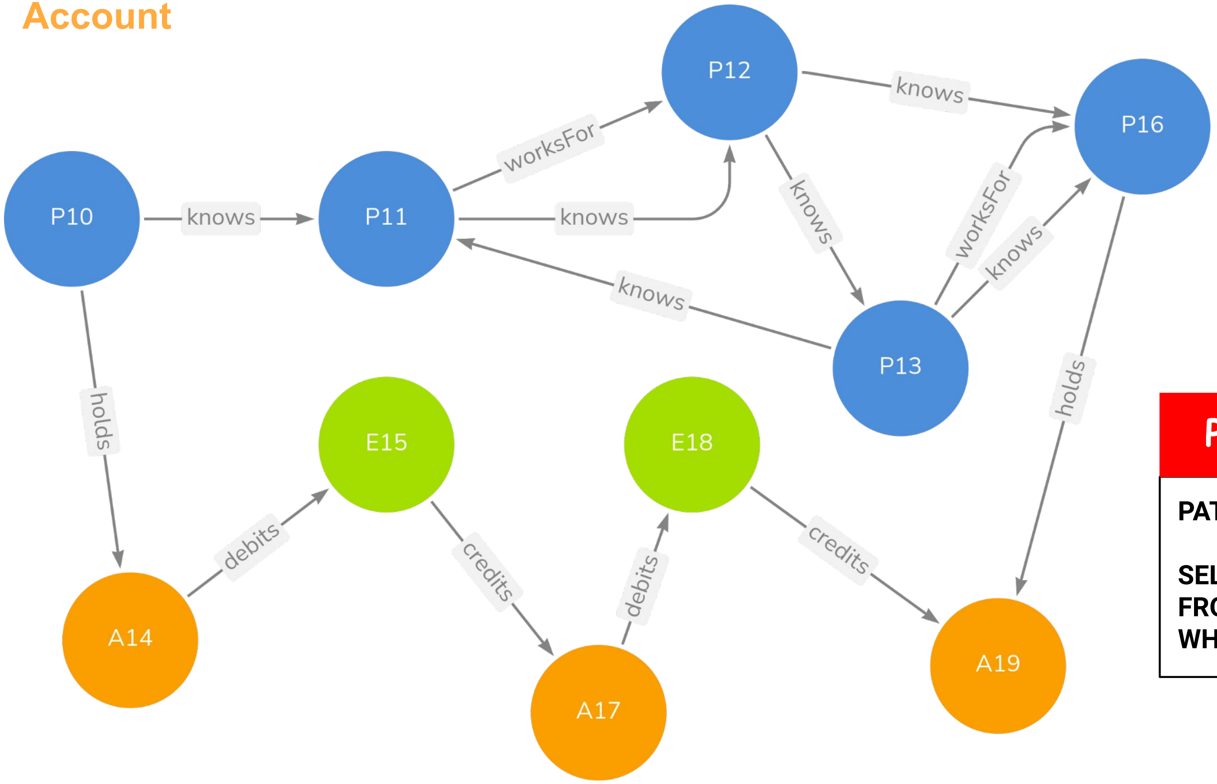


V	$L_{in}(v)$	$L_{out}(v)$
1		
2	(1, {r})	(1, {b})
3	(1, {g})	(1,{r,b}), (2,{r})
4	(1,{r,g}), (1,{r,b}), (2,{g}), (3,{b})	
5	(1,{r,g}), (2,{g})	(4,{g})
6	(1,{r}), (2,{r,g}), (5,{r})	
7	(1,{r})	

[Pen20] Y. Peng et al. Answering billion-scale label-constrained reachability queries within microsecond. Proc. VLDB Endow. 13(6): 812-825 (2020)
[Aki13]. Akiba et al. Fast exact shortest-path distance queries on large networks by pruned landmark labeling. SIGMOD Conference 2013: 349-360


GO
LCR Queries
RLC Queries

Person
External Entity
Account



Money laundering analysis:

Do accounts 14 and 19 have the repeated outside-inside money transferring pattern?

PGQL

```
PATH out_in AS (:Person) -[:debits-]> (:EnternalEntity)
                    -[:credits]-> (:Person)
SELECT *
FROM MATH (s) -/:out_in+/-> (t)
WHERE ID(s) = 14 AND ID(t) = 19
```

RLC (recursive label-concatenated) queries

- CL (concatenated labels)
  - $(l_1, \ldots, l_k)^+$, where labels are **concatenated**
- RLC query (s, t, CL) [Zha22]
  - Checking whether s reaches t
  - Checking whether the path **matches the given CL pattern**
- Boolean query
  - Returning either True or False
- Supported languages
  - SPARQL
  - PGQL

- Path semantics
  - **Arbitrary paths**
- RLC queries appear quite often in timeout query logs [Bon19]

[Zha22] C. Zhang et al. A Reachability Index for Recursive Label-Concatenated Graph Queries. CoRR abs/2203.08606 (2022)
[Bon19] A. Bonifati et al. Navigating the Maze of Wikidata Query Logs. WWW 2019: 127-138

A Reachability Index for Recursive Label-Concatenated Graph Queries

Chao Zhang*, Angela Bonifati*, Hugo Kapp†, Vlad Ioan Haprian† and Jean-Pierre Lozi†
*Lyon 1 University, Lyon, France
†Oracle Labs, Zürich, Switzerland
{chao.zhang, angela.bonifati}@univ-lyon1.fr, {hugo.kapp, vlad.haprian, jean-pierre.lozi}@oracle.com

***Abstract*—Reachability queries checking the existence of a path from a source node to a target node are fundamental operators for querying and processing graph data. Current approaches for index-based evaluation of reachability queries either focus on plain reachability or constraint-based reachability with only alternation of labels. In this paper, for the first time we study the problem of index-based processing for recursive label-concatenated reachability queries, referred to as RLC queries. These queries check the existence of a path that can satisfy the constraint defined by a concatenation of at most k edge labels under the Kleene plus. Many practical graph database and network analysis applications exhibit RLC queries. However, their evaluation remains prohibitive in current graph database engines.**

We introduce the RLC index, the first reachability index to efficiently process RLC queries. The RLC index checks whether the source vertex can reach an intermediate vertex that can also reach the target vertex under a recursive label-concatenated constraint. We propose an indexing algorithm to build the RLC index, which guarantees the soundness and the completeness of query execution and avoids recording redundant index entries. Comprehensive experiments on real-world graphs show that the RLC index can significantly reduce both the offline processing cost and the memory overhead of transitive closure, while improving query processing up to six orders of magnitude over online traversals. Finally, our open-source implementation of the RLC index significantly outperforms current mainstream graph engines for evaluating RLC queries.

***Index Terms*—reachability index, graph query, graph databases, RLC queries**

I. INTRODUCTION

Graphs have been the natural choice of data representation in various domains [1], *e.g.*, social, biochemical, fraud detection and transportation networks, and reachability queries are fundamental graph operators [2]. Plain reachability queries check whether there exists a path from a source vertex to a target vertex, for which various indexing techniques have been proposed [3]–[18]. To facilitate the representation of different types of relationships in real-world applications, *edge-labeled graphs* and *property graphs*, where labels can be assigned to edges, are more widely adopted nowadays than unlabeled graphs. Such advanced graph models allow users to add path constraints when defining reachability queries, which play a key role in graph analytics. However, current index-based approaches focus on constraint-based reachability with only alternation [19]–[23]. In this paper, we consider for the first time reachability queries with a path constraint corresponding to a

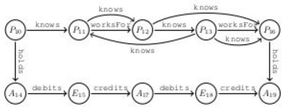


Fig. 1. A social and professional network on which RLC queries are instantiated.

concatenation of edge labels under the Kleene plus, referred to as *recursive label-concatenated queries* (RLC queries). RLC queries call for a novel indexing technique due to inherently different path constraints compared to either plain reachability queries or alternation-based reachability, respectively. To further motivate RLC queries, we present a running example in the following.

Running example. Figure 1 shows a property graph inspired by a real-world use case encoding an interleaved social and professional network along with information of bank accounts of persons. RLC queries can be used to detect fraud and money laundering patterns among financial transactions. For instance, the query $Q1(A_{14}, A_{19}, (\texttt{debits}, \texttt{credits})^+)$ checks whether there is a path from account A_{14} to A_{19} such that the label sequence of the path is a concatenation of an arbitrary number (one or more) of occurrences of (debits, credits), which can lead to detect suspicious patterns of money transfers between these accounts. The RLC query $Q1((A_{14}, A_{19}, (\texttt{debits}, \texttt{credits})^+)$ evaluates to *true* because of the existence of the path (A_{14}, debits, E_{15}, credits, A_{17}, debits, E_{18}, credits, A_{19}). Another example is $Q2(P_{10}, P_{13}, (\texttt{knows}, \texttt{knows}, \texttt{worksFor})^+)$ that evaluates to false because there is no path from P_{10} to P_{13} satisfying the constraint.

RLC queries are also frequently occurring in real-world query logs, *e.g.*, Wikidata Query Logs [24], which is the largest repository of open-source graph queries (of the order of 500M queries). In particular, RLC queries often time out in these logs [24] thus showing the limitations of graph query engines to efficiently evaluate them. Moreover, Neo4j (v4.3) [25] and TigerGraph (v3.3) [26], two of the mainstream graph data processing engines, do not yet support RLC queries in

1

https://arxiv.org/abs/2203.08606

g-rpqs/**rlc-index**

This repository provides the RLC index, a reachability index for processing graph queries with a concatenation of edge labels under...

1 Contributor · 0 Issues · 3 Stars · 0 Forks

https://github.com/g-rpqs/rlc-index

UNIVERSITÉ DE LYON

Université Claude Bernard Lyon 1

Oracle Labs

PGX

Challenges for reachability indexes with path constraints

1. Limited resources
   - Partial index + Guided online search
2. Beyond static graphs
   - Dynamic graphs
     - Append-only graphs
     - Fully dynamic graphs
   - Streaming graphs [Pac20]
3. Distributed graphs

4. More regular expressions [Bon19]
5. Upper and lower bound of hops
6. REM [Lib12]: topology + data
7. Temporal graph query with time interval [Ros22]

[Pac20] A. Pacaci et al. Regular Path Query Evaluation on Streaming Graphs. SIGMOD Conference 2020: 1415-1430
[Bon19] A. Bonifati et al. Navigating the Maze of Wikidata Query Logs. WWW 2019: 127-138
[Lib12] L. Libkin et al. Regular path queries on graphs with data. ICDT 2012: 74-85
[Ros22] C. Rost et al. Distributed temporal graph analytics with GRADOOP. VLDB J. 31(2): 375-401 (2022)

Thank you and Q&A