

Landmark indexing for evaluation of label-constrained reachability queries

Lucien Valstar[†], George Fletcher[†], Yuichi Yoshida[‡]

[†]TU Eindhoven (Netherlands),

[‡]National Institute of Informatics
and Preferred Infrastructure, Inc. (Japan)

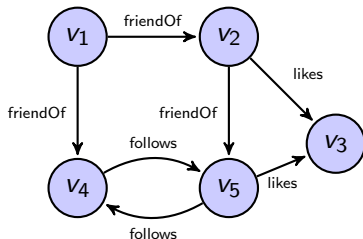
SIGMOD 2017

Chicago, 16 May 2017

Labeled networks

Big **graph** data sets are ubiquitous

- ▶ social networks (e.g., LinkedIn, Facebook)
- ▶ scientific networks (e.g., Uniprot, PubChem)
- ▶ knowledge graphs (e.g., DBPedia, MS Academic Graph)
- ▶ transportation and utility networks
- ▶ ...



Focus is on “**things**” (i.e., nodes, vertices) and their **relationships** (i.e., **labeled** directed edges)

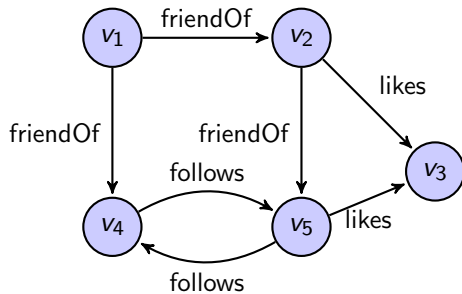
Label-constrained reachability queries on networks

We study **Label-Constrained Reachability (LCR) Queries** on networks:

Given vertices s and t of labeled graph G and a subset L of the set of all edge labels \mathcal{L} of G , determine whether or not there is a path from s to t using only edges with labels in L .

When such a path exists, we denote this by $s \overset{L}{\rightsquigarrow} t$.

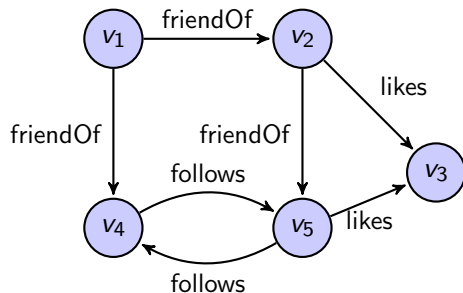
Label-constrained reachability queries on networks



Example. The query $(v_1, v_5, \{\text{friendOf}\})$ is true.

The query $(v_1, v_3, \{\text{friendOf}\})$ is false.

Label-constrained reachability queries on networks



Example. The query $(v_1, v_5, \{\text{friendOf}\})$ is true.

The query $(v_1, v_3, \{\text{friendOf}\})$ is false.

LCR Queries

- ▶ Natural generalization of reachability queries.
- ▶ An important fragment of the language of [regular path queries](#).
- ▶ Implemented in W3C's SPARQL 1.1, Neo4j's Cypher, and Oracle's PGQL.

LCR queries: current evaluation solutions

Despite the importance of LCR queries, current solutions do not scale to large graphs occurring in practice.

There are two approaches to solving LCR queries: **exhaustive search** using state-of-the-art methods such as direction-optimizing BFS (DBFS)

- ▶ Beamer et al. *Scientific Programming* 21, 2013

or graph **indexing** for accelerated search

- ▶ Jin et al. *SIGMOD* 2010
- ▶ Bonchi et al. *EDBT*, 2014
- ▶ Zou et al. *Information Systems* 40, 2014.

LCR queries: our contributions

Our contributions. New indexing methods for LCR queries exploiting landmark vertices.

- ▶ Scales to **orders of magnitude larger graphs** than current indexing methods.
- ▶ Up to **orders of magnitude faster query evaluation** than current solutions.
- ▶ Our implementation is publicly available as open-source at <https://github.com/DeLaChance/LCR>

Landmark indexing for LCR: naive solution

Naive Idea (FULL-LI)

Given a graph (V, E, \mathcal{L}) , for each vertex $v \in V$, store in an index the pairs $\{(w, L) \mid w \in V, L \subseteq \mathcal{L}, \text{ and } v \overset{L}{\rightsquigarrow} w\}$.

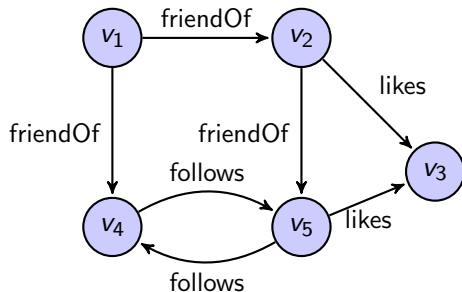
Landmark indexing for LCR: naive solution

Naive Idea (FULL-LI)

Given a graph (V, E, \mathcal{L}) , for each vertex $v \in V$, store in an index the pairs $\{(w, L) \mid w \in V, L \subseteq \mathcal{L}, \text{ and } v \overset{L}{\rightsquigarrow} w\}$.

Given a query (s, t, L) , just check whether or not (t, L) is in the index for s .

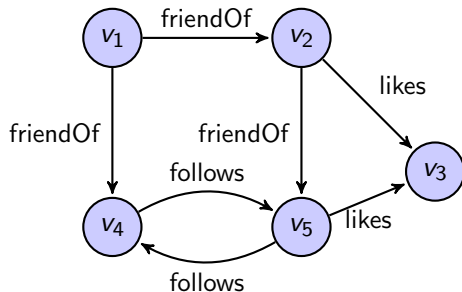
Landmark indexing for LCR: naive solution



Example. The FULL-LI index entry for v_2 :

$(v_3, \{\text{likes}\})$,
 $(v_3, \{\text{friendOf}, \text{likes}\})$,
 $(v_3, \{\text{friendOf}, \text{follows}, \text{likes}\})$,
 $(v_4, \{\text{friendOf}, \text{follows}\})$,
 $(v_5, \{\text{friendOf}\})$,
 $(v_5, \{\text{friendOf}, \text{follows}\})$.

Landmark indexing for LCR: naive solution



Example. The FULL-LI index entry for v_2 :

$(v_3, \{\text{likes}\})$,
 $(v_3, \{\text{friendOf}, \text{likes}\})$,
 $(v_3, \{\text{friendOf}, \text{follows}, \text{likes}\})$,
 $(v_4, \{\text{friendOf}, \text{follows}\})$,
 $(v_5, \{\text{friendOf}\})$,
 $(v_5, \{\text{friendOf}, \text{follows}\})$.

Naive Idea (FULL-LI)

- ▶ Excellent query performance.
- ▶ Does not scale to large graphs.

Landmark indexing for LCR: selective landmarking

Landmark Index (LI)

Only build indexes for a select small number of **landmark** vertices

- ▶ e.g., top k vertices of highest degree

Furthermore, only store entries (w, L) such that L is a **minimal label set** connecting v to w

- ▶ that is, there is no L' strictly contained in L such that $v \overset{L'}{\rightsquigarrow} w$.

Landmark indexing for LCR: selective landmarking

Landmark Index (LI)

Only build indexes for a select small number of **landmark** vertices

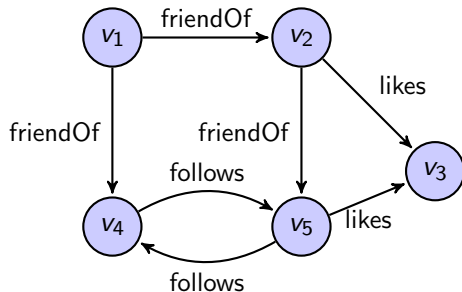
- ▶ e.g., top k vertices of highest degree

Furthermore, only store entries (w, L) such that L is a **minimal label set** connecting v to w

- ▶ that is, there is no L' strictly contained in L such that $v \xrightarrow{L'} w$.

Given a query (s, t, L) , perform BFS from s only using edges with labels in L . When we hit a landmark vertex, we use its index to obtain the answer immediately.

Landmark indexing for LCR: selective landmarking

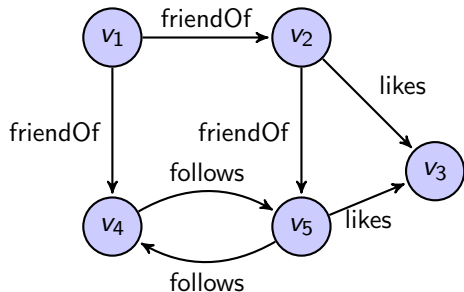


Example. The LI index entry for v_2 :

$(v_3, \{\text{likes}\}),$
 $(v_4, \{\text{friendOf}, \text{follows}\}),$
 $(v_5, \{\text{friendOf}\}).$

Half as many entries as FULL-LI entry for v_2 .

Landmark indexing for LCR: selective landmarking



Example. The LI index entry for v_2 :

$(v_3, \{\text{likes}\}),$
 $(v_4, \{\text{friendOf}, \text{follows}\}),$
 $(v_5, \{\text{friendOf}\}).$

Half as many entries as FULL-LI entry for v_2 .

Landmark index (LI)

- ▶ Balances space/time.
- ▶ Can significantly reduce index size.
- ▶ Still obtain the benefits of accelerated search.

Landmark indexing for LCR: extended indexing

Extended Landmark Index (LI⁺)

Two extensions to make LI more efficient.

(1) It may take a long time before finding a landmark. We can remedy this by building an **incomplete index for non-landmarks**: for each non-landmark v , we insert a small number of entries (v', L) where v' is a landmark and $v \xrightarrow{L} v'$.

These provide shortcuts to landmarks during search.

Landmark indexing for LCR: extended indexing

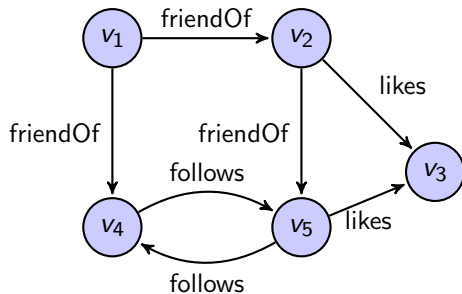
Extended Landmark Index (LI⁺)

Two extensions to make LI more efficient.

(2) There is a strong asymmetry in evaluation of true- and false-queries. A true-query can stop after finding a landmark, whereas a false-query often needs to explore larger parts of the graph.

To remedy this, we can maintain for each landmark v and label set L the “reachable-by” set $R_L(v) = \{w \in V \mid v \overset{L}{\rightsquigarrow} w\}$.

Landmark indexing for LCR: extended indexing



Example.

$$R_{\{\text{friendOf}\}}(v_1) = \{v_2, v_4, v_5\}.$$

Landmark indexing for LCR: extended indexing

Extended Landmark Index (LI⁺)

Two extensions to make LI more efficient.

(2, cont.) During evaluation of query (s, t, L) , suppose we have found $s \overset{L}{\rightsquigarrow} v$ and $v \not\overset{L}{\rightsquigarrow} t$, for some landmark v .

Then, for every $w \in R_L(v)$, we must have $w \not\overset{L}{\rightsquigarrow} t$.

Hence, we can **mark and never visit any vertex of $R_L(v)$** during the rest of the search.

Landmark indexing for LCR: extended indexing

Extended Landmark Index (LI⁺)

Two extensions to make LI more efficient.

(2, cont.) During evaluation of query (s, t, L) , suppose we have found $s \xrightarrow{L} v$ and $v \not\xrightarrow{L} t$, for some landmark v .

Then, for every $w \in R_L(v)$, we must have $w \not\xrightarrow{L} t$.

Hence, we can **mark and never visit any vertex of $R_L(v)$** during the rest of the search.

For practical purposes, we only keep a subset of the $R_L(v)$'s, and only for relatively small label sets.

Settings

- ▶ Linux server with 258GB of memory and a 2.9GHz 32-core processor.
- ▶ Fourteen real networks, ranging from social networks to biological networks.
- ▶ The number of landmarks is $1250 + \sqrt{n}$, where n is the number of vertices. The budget for non-landmarks is 20.
- ▶ Generated sets of 1000 true-queries and 1000 false-queries.

Experimental study

Settings

- ▶ Linux server with 258GB of memory and a 2.9GHz 32-core processor.
- ▶ Fourteen real networks, ranging from social networks to biological networks.
- ▶ The number of landmarks is $1250 + \sqrt{n}$, where n is the number of vertices. The budget for non-landmarks is 20.
- ▶ Generated sets of 1000 true-queries and 1000 false-queries.

We report here highlights of results on [indexing costs](#) and [query performance](#).

See our paper for full details and also details of performance on synthetic graphs where we study the [impact of graph density, label set size, and graph structure](#).

Experimental study: indexing costs

Dataset	LI		LI ⁺		FULL-LI		ZOU	
	IT	IS	IT	IS	IT	IS	IT	IS
robots	0.1	5	0.1	5	0.1	5	9	5
advogato	4	135	3	131	7	369	11,867	369
epinions	272	2,903	205	2,091	-	-	-	-
NotreDame	242	2,424	193	1,895	-	-	-	-
BioGrid	58	1,410	50	1,302	36	3,207	-	-
webGoogle	5,887	33,931	5,665	33,497	-	-	-	-
Youtube	3,121	336	2,841	300	-	-	-	-
socPokec	9,762	77,155	9,461	75,698	-	-	-	-
wikiLinks(fr)	24,873	98,125	25,641	103,414	-	-	-	-

Number of edges: **robots** 2.9k, **advogato** 51k, **epinions** 840k, **NotreDame** 1.4M, **BioGrid** 1.5M, **webGoogle** 5.1M, **Youtube** 10.7M, **socPokec** 30.6M, **wikiLinks(fr)** 102.3M.

Experimental study: query performance

Dataset	true			false		
	LI	LI ⁺	DBFS (μ s)	LI	LI ⁺	DBFS (μ s)
robots	17.63	17.63	1.77	6.95	6.95	0.70
advogato	84.25	93.08	20.6	3.33	1.89	0.67
epinions	69.18	52.10	106	0.00	0.58	1.91
NotreDame	22.27	7.27	159	5.17	49.44	555
BioGrid	16.98	20.79	848	0.18	37.95	709
webGoogle	338.26	184.52	1,340	0.00	0.57	9.65
Youtube	16.10	21.10	2,000	0.53	12.19	3,880
socPokec	9.20	10.81	1,290	0.00	0.25	39.8
wikiLinks(fr)	54.53	44.54	3,120	0.00	0.42	38.8

General observations.

- ▶ For true-queries, LI and LI⁺ are always advantageous, accelerating query evaluation up to two orders of magnitude over state-of-the-art search methods.
For false-queries, LI⁺ was within the same order of magnitude or better for the majority of cases.
 - ▶ Often we found that search failure occurred much closer to the target, to the advantage of DBFS.
This indicates the need in future work to study direction-optimizing variants of our solutions.

General observations.

- ▶ For true-queries, LI and LI⁺ are always advantageous, accelerating query evaluation up to two orders of magnitude over state-of-the-art search methods.
For false-queries, LI⁺ was within the same order of magnitude or better for the majority of cases.
 - ▶ Often we found that search failure occurred much closer to the target, to the advantage of DBFS.
This indicates the need in future work to study direction-optimizing variants of our solutions.
- ▶ LI and LI⁺ are the only indexing strategies which can handle large graphs, up to four orders of magnitude larger than current indexing strategies.

Concluding remarks

Our contributions: New landmark-based indexing solutions for scalable evaluation of LCR queries, scaling to orders-of magnitude larger graphs and orders of magnitude faster evaluation time.

Concluding remarks

Looking ahead:

- ▶ finer analysis of the impact of graph topology and complexity on the performance and further optimization of our solutions, e.g., graphs of bounded treewidth.
- ▶ study of landmark-based evaluation methods for extensions to the class of LCR queries.
- ▶ the study of landmark indexing in multi-core environments.
- ▶ study of alternative search strategies for improving performance on false queries.
- ▶ applications of our indexes for evaluation of practical query languages such as SPARQL 1.1 and openCypher.

Landmark indexing for evaluation of label-constrained reachability queries

Lucien Valstar, George Fletcher, and Yuichi Yoshida

TU Eindhoven (Netherlands)
National Institute of Informatics
and Preferred Infrastructure, Inc. (Japan)

<https://github.com/DeLaChance/LCR>

Thank you!