All Labels Combined Method for Label-Constrained Reachability Queries

Mohammad Sadegh Najafi
Dept. of Computer Science
Seoul National University
najafi@tcs.snu.ac.kr

Chris Hickey
Dept. of Cognitive Science
Seoul National University
chris.hickey@ucdconnect.ie

October 2, 2019

Abstract

Real world network graphs such as social networks, semantic verbs and citation networks can often containing varying different edge labels connecting vertices. For example, in the context of social networks, nodes can be connected via 'likes', 'comments', 'friendship' etc. What is the fastest, most space efficient way to query whether a path exists between two vertices in a network, using only a subset of vertices? This present research proposal aims to contribute to this question by comparing performance of the current state of the art label-constrained reachability query 'Landmark Indexing', to a label-constrained reachability query which we have developed ourselves. These algorithms will be compared both in terms of time complexity and memory space efficiency. We expect that the current state-of-the art 'Landmark Indexing' algorithm may perform better than the present studies proposed algorithm in terms of time-complexity for some queries. However, we also expected that our proposed algorithm will perform better than 'Landmark Indexing' in terms of memory space efficiency and in terms of time complexity for certain queries such as queries that return $false$.

1 Introduction

As graph databases and real world network graphs continue to increase in popularity and expand in size, the ability to efficiently and reliably query information from these graphs will also increase in importance. One such family of queries which are garnering notable amounts of research of late are *reachability queries*. Reachability queries work as follows. Given any two nodes in a graph v_1 and v_2, does a path exist the two nodes. As outlined by [7] in their survey of reachability queries, these type of queries are used across a wide range of use cases including web usage mining, web site analysis, and biological network analysis. For example in bioinformatics, graph vertices may be either molecules, reactions, or interactions in living

cells. Edges in such a biological network will represent how these various biological elements interact with each other. In this context, reachablity queries help biologists determine which genes are influenced (either directly or indirectly) by a given molecule.

Label-constrained reachability (LCR) queries represent a subset of reachability queries that have been attracting much interest of late. This subset of reachability queries is concerned with finding a path between two vertices in a graph, using a only subset of the labels that exist between vertices in the graph. For example, in the context of a social network, people may be connected with each other via 'likes', 'comments', or 'friendship.' In this context, an example of a graph reachability query would be to investigate whether there was a path between any two people (vertices) in the social network. On the other hand, an example of a label constrained reachability query would be to investigate whether there exists a path between two people in the social network via only 'likes' and 'comments'.

As highlighted by the Zho et al.[8], there are two extreme solutions to answering any reachability query. They claim that the first approach is to create the transitive closure of a graph across all potential subsets of edge labels. This would enable all reachability queries to be answered extremely efficiently time wise. The other possible approach is to perform a Breadth First Search (BFS) over the graph in response to any given query at runtime. As pointed out by Zho and colleagues[8] Both these approaches on their own are not feasible in real world graphs at scale. Even in the context of standard reachability queries, the closure approach would require $O([V]^2)$ memory space to store each transitive closure to respond to queries. Furthermore, the BFS approach would require $O([V])$ time complexity in order to respond to each individual query. As such, these two methods can not work on real world massive graphs on their own as either their time or space complexity is too large to work in practice. Any practical reachability query solution should aim to strike some sort of compromise between the two solutions.

The Landmark Indexing algorithm proposed recently by Valstar, Fletcher and Yuichi in 2017[5] aims to provide such a compromise for solving LCR queries. Landmark Indexing works by constructing indices for a small group of vertices. These vertices are subsequently referred to as "Landmarks". Landmarks are chosen based on their degree, with larger degree nodes being more likely to be chosen as landmarks. Therefore, given a graph G=(V, E, L) (see Table 1 for notation meaning), for each landmark vertex v, an index us stored of $(w, L') \in V \times 2^{\ell}$ if there exists an L'-path from v to w. For queries where the source vertex is a landmark, the query can be responded to by simply checking the index. Otherwise, a BFS is conducted across the restricted labeled graph until a landmark is reached. In this regard, the Landmark Indexing algorithm can be seen as somewhat of a compromise between time efficiency and memory efficiency approaches.

We are proposing an alternative method for responding to LCR queries. We refer to this approach as the *All Labels Combination* (ALC) algorithm. We shall explain this new proposed methodology in the 'Proposed Method' section of the paper.

The main problem being solved by the present research proposal is as follows:

- GIVEN: s and t of graph G and an L' subset of L.
- FIND: $true/false$ whether or not there exists a path from s to t in G using only edges

with labels in L', using both the state of the Landmark Indexing algorithm and the newly proposed ALC algorithm.
- to DETERMINE: which algorithm performs best given a specific graph G:

This is an important problem, because as discussed before, reachability queries have important use cases ranging greatly in diversity from biological networks to social networks. However depending on the size of the graph, different approaches may be more or less feasible. The best, most efficient way to run a LCR query on a graph remains an open question in computer science. As such, testing different LCR algorithms on graphs of different sizes is vital in order to understand which queries work best on small, medium and large sized graphs with varying characteristics.

The contributions of this project are the following:

- We aim to offer an alternative ALC method to the current state of the art Landmark Indexing algorithm for performing LCR queries.
- In terms of the memory space/time complexity trade off, we expect the proposed algorithm to be more space efficient than the Landmark Indexing approach. However we will also work to provide speeds that are comparable to Landmark Indexing
- We aim for our proposed ALC method to be more suitable for queries on extremely large graphs, where the memory demands of the Landmark Indexing approach may make this approach unfeasible. We also expect the ALC algorithm to perform better than landmark indexing for queries that return $false$ in all cases.

Symbol	Definition
G	Graph being queried
V	A finite set of vertices in the graph
E	The set of all edges (v, w, L') where an edges exists from v to w with some label L'
L	Set of all labels in the graph
ℓ	Length of L
L'	Any subset of labels from L

Table 1: Symbols and definitions

2 Survey

Next we list the papers that each member read, along with their summary and critique.

2.1 Papers read by Mohammad Sadegh Najafi

First and foremost, apart from the Landmark indexing, which its evaluation is the main focus of this work, we largely review all recent developments in the Label Constrained Reachability area.

2.1.1 Shortest Path Label-Constrained [2]

- *Main idea*: the main focus of this study is to present label-constrained shortest path(LCSP) method in order to discover the shortest path between the source and target only with constrained labels. with regard to the suggested method, two approaches with an approximation of finding the precise shortest path while false negative is possible.
- *Use for our project*: the paper proposes a recent improvement on Label Constrained Reachability queries. LCSP provides a general methodology for processing reachability queries. The author proposes two methods to evaluate LCSP queries. The proposed methods are an approximation of the actual distance.
- *Shortcomings*: due to the fact that the final result of the evaluation is an approximation of the actual distance, they are not useful as well as we do not consider this work on our discussion.

2.1.2 Tree-based Index Framework [3]

- *Main idea*: bauilding a full transitive closure (TC) of the underlying graph to answer LCR queries. A tree-based index framework which contains a spanning tree T and a partial transitive closure PT of the graph. PT and T have the information to derive full TC.
- *Use for our project*: the Landmark Indexing (LI) method has some properties in common with Jin's method. the better understanding the structure of this work would help us accurately analyze properties of LI. LI does not construct a full TC and it attempts to keep the balance between BFS and constructing full TC. As a result, the running time for query evaluation declines.
- *Shortcomings*: there is a real downside for Jin's method. The offered method is not practical for dense graphs owing to the comparatively small size of spanning-tree T in comparison with the whole graph.

2.1.3 Approximate Regular-simple-path Reachability [6]

- *Main idea*: all recent techniques have limitations to some extent form network growth to efficiency for a limited number of labels. They generalize regular simple path queries for LCR. They aim to develop an almost optimal index-free to evaluate LCR.
- *Use for our project*: this novel approach evaluates Landmark Indexing as well as propose a state-of-the-art model to dynamically adapt the method to the network growth. Our expectation for the ACL is the more comparable time complexity compares to Landmark Indexing. With regards to this innovative approach, we can allocate further researches to decrease the time complexity of ACL.
- *Shortcomings*: their method has improved several downsides of previous works including adaptability to network growth and independence from pre-computing processes.

2.2 Papers read by Chris Hickey

The first paper I read was a general introduction to reachability queries provided by Yu and Cheng [7].

- *Main idea*: This is simply an introductory chapter into reachability queries in general. It also offers a very clear definition of the problem: Given directed graph $G = (V, E)$ with n nodes and m edges, you are asked to find a path between two nodes v and u. If a path exists between the two nodes, the query will return true, otherwise it will return false. This can be redefined as, given two nodes v and u, and the edge transitive closure TC of graph G, is $(u, v) \in TC$. The rest of the chapter was dedicated to current best approaches on how to solve reachability queries in unlabeled constrained directed graphs.
- *Use for our project*: as somebody with no previous experience in reachability queries, this introductory chapter gave me an excellent insight into why reachability queries in general are important in domains ranging from social networks to bioinformatics. The main takeaway I had from this chapter were the different time complexities, index construction time complexities and index size of the current best approaches to solving standard reachability queries. These complexities really drove home the point that reachability queries are all about striking a compromise between time complexity and space complexity. These can range drastically between a transitive closure approach, where query time is $O(1)$ but memory space is $O(n^2)$, to a 3-Hop Cover algorithm where query time is $O(logn + k)$ but memory space is $O(nk)$.
- *Shortcomings*: The authors fail to give examples as to which approaches are best suited to which graphs. For example, the authors stress that 2-hop and 3-hop approaches are difficult to compute. I presume this means such approaches are not suitable to rapidly changing graphs like social networks. I would have preferred if the authors gave specific examples of which reachability query approaches were best suited to different types of graphs. This question will be key to our research.

The second paper I read was about Land Mark Indexing as a solution to LCR queries[5]. This will be used as the benchmark algorithm for our project.

- *Main idea*: by selecting the vertices with the largest degree, and constructing an index such that given a query (s, t, L'), the query can be answered immediately if s is a landmark, the majority of label-constricted queries can by answered. This index is constructed by for every landmark index v, storing every pair $(u, L') \in V \times 2^{\ell}$, where L' is all possible subset labels for the label super set L. For queries where the source vertex is not a landmark, an optimized breath-first search algorithm is used until a landmark is found, then the query is responded to.
- *Use for our project*: As one of the more recent development in LCR queries, this serves as an excellent benchmark for our proposed algorithm. Indeed the entire goal of our project is to offer an improvement on this algorithm, at least for certain graphs.
- *Shortcomings*: One glaring issue with landmark indexing is it's performance for non landmarked nodes. The authors note that especially for LCR queries on unlandmarked

nodes where the answer is 'false', the algorithm performs no better than BFS. Given the indexing memory space taken up by the landmarks, this algorithm represents a 'worst of both worlds' scenario for LCR queries which return false, performing bad both in terms of speed and memory space.

The third paper I read focused on attempts to optimize BFS [1]. BFS plays an instrumental role both in Landmark Indexing, and our proposed algorithm.

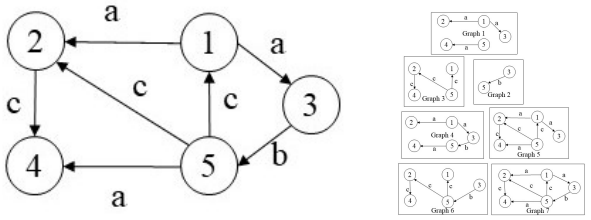
- *Main idea*: The authors extend the standard top down BFS as follows. In standard BFS, starting from the source vertex, all of the vertices at the same depth are visited before any vertex at the next next depth is visited. The authors combine this approach with a novel top up approach, where unvisited vertices attempt to find any parent among its neighbors which are still currently in the active 'frontier' of possible intermediary nodes between the source vertex and the target vertex. Experiments showed this approach to be between 2-5 times faster than normal BFS on real world social network graphs.
- *Use for our project*: Both our proposed ALC approach and the Landmark Indexing algorithms depend on BFS in some form or another. As such, understanding how to best implement BFS is vital to getting the best performance out of both algorithms
- *Shortcomings*: The authors explan why this approach is useful for 'small world' phenomenon graphs, like social networks. I would be interested to know if there are graphs where this hybrid Directional BFS approach is less useful than conventional BFS.

3 Proposed Method

The main contribution of the paper is to offer an alternative method Landmark Indexing for LCR queries. The alternative method will work as follows: Given graph G=(V, E, L), the proposed ALC methodology will construct multiple graphs of all possible combinations of labels. This leads to the construction of $2^l - 1$ unlabeled subgraphs. For example, if the total set of labels for a graph is $L = \{a, b, c\}$ then the ALC algorithm will first take all possible label combinations $\{a\}$,$\{b\}$,$\{c\}$,$\{a, b\}$,$\{a, c\}$,$\{b, c\}$ $\{a, b, c\}$, creating graphs based exclusively on edges in the label subsets. Then, given A query (s, t, L'), the pre computed subgraph of G corresponding to the the subset of labels specified in L' are first brought forward. BFS is then computed over this subgraph in order to establish whether a path exists between source node s and destination t.

We expect the propose ALC algorithm to have the following contrasts when compared the Landmark Indexing algorithm:

- *Memory Space*: Our proposed memory space algorithm will be more efficient in terms of memory space than Landmark Indexing
- *More suitable for sparse networks*: Unlike Landmark Indexing, the propose ALC algorithm treats all nodes equally. As such, it will perform equally well on sparse graphs or on graphs where no vertex has a notably higher degree than any other vertex.



(a) (b)

Figure 1: An input labeled graph (a) and the unlabled graphs that would be created via ALC (b)

- *Speed Performance*: The Landmark Indexing algorithm will perform faster than the ALC algorithm in cases where the queried vertex is a landmark vertex. However, if the query returns false, then we expect the ALC algorithm to perform better than Landmark Indexing.

4 Experiments

We intend to compare the proposed ACL algorithm to current state of the art Landmark Indexing algorithm across varying graph sizes with various different labels. In order to generate synthetic graphs to test both algorithms against, we intend to use SNAP [4]. We are also currently searching for real world graphs for comparing both Landmark Indexing and ALC.

The Landmark Indexing algorithm and SNAP library for graph creation are both written in C++. As such, we the proposed ACL algorithm will also be written in C++.

Queries will take the form of a source vertex s, a destination vertex d, and a subset of the total labels L' belonging to the graph. These queries will return true if a path exists between s and d using just the subset of labels in L'.

Queries will be judged on two criteria,

- *Speed*: Time of response in milliseconds
- *Memory*: Amount of memory space required to service queries.

5 Conclusions

Through this initial survey and research we have come to the following conclusions

- Reachability queries are important in a wide variety of real life graphs.
- The question of how to best service reachability queries in the context of Label Restrained Reachability queries remains very much an open question to this day.
- A current state of the art method for LCR queries is Landmark Indexing.

  - Landmark Indexing works well when a LCR query is being made from a landmark indexing node.
  - When not querying a landmark indexed node, performance can be extremely slow. Indeed when making queries that return false, performance will be same as BFS.
  - The memory space required by Landmark Indexing is quite large, and increases with the number of vertices marked as designated landmarks.
- The proposed All Combination Labels algorithms improves on Landmark Indexing in the following way:
  - Being more practical for large scale graphs in terms of memory space usage.
  - Speed up LCR that respond *true* in some case, and $false$ in all cases

References

[1] Scott Beamer, Krste Asanović, and David Patterson. Direction-optimizing breadth-first search. *Scientific Programming*, 21(3-4):137–148, 2013.

[2] Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Antti Ukkonen. Distance oracles in edge-labeled graphs. In *EDBT*, pages 547–558, 2014.

[3] Ruoming Jin, Hui Hong, Haixun Wang, Ning Ruan, and Yang Xiang. Computing label-constraint reachability in graph databases. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 123–134. ACM, 2010.

[4] Jure Leskovec and Rok Sosič. Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):1, 2016.

[5] Lucien DJ Valstar, George HL Fletcher, and Yuichi Yoshida. Landmark indexing for evaluation of label-constrained reachability queries. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 345–358. ACM, 2017.

[6] Sarisht Wadhwa, Anagh Prasad, Sayan Ranu, Amitabha Bagchi, and Srikanta Bedathur. Efficiently answering regular simple path queries on large labeled networks. *Age*, 3:v7, 2019.

[7] Jeffrey Xu Yu and Jiefeng Cheng. *Graph Reachability Queries: A Survey*, pages 181–215. Springer US, Boston, MA, 2010.

[8] Xu K. Yu J.X. Chen L. Xiao Y. Zou, L. and D." Zhao. Efficient processing of label-constraint reachability queries in large graphs. *Information Systems*, 40:47–66, 2014.

A Appendix

A.1 Labor Division

The team performed the following tasks

- **Mohammad**:
  - Implementation of ALC
  - Synthetic Graph Creation
  - Study For Further Developments
- **Chris**:
  - Provide Assistance and Test Cases
  - Provide the Landmark indexing implementation for comparison
  - Research For Potential Alternative LCR queries
- **Both**:
  - Experiments using ACL and Landmark Indexing
  - Implement Further Developments

A.2 Full disclosure wrt dissertations/projects

Mohammad: Mohammad has attempted to create an algorithm for LCR before. However performance was too slow to be of much practical use.

Chris: He is not doing any project or dissertation related to this project.

Contents