Hop-Constrained *s-t* Simple Path Enumeration in Billion-Scale Labelled Graphs

Xia Li[1], Kongzhang Hao[1(✉)], Zhengyi Yang[1], Xin Cao[1], Wenjie Zhang[1], Long Yuan[2], and Xuemin Lin[3]

[1] The University of New South Wales, Sydney, Australia
{xia.li,k.hao,zhengyi.yang,xin.cao,wenjie.zhang}@unsw.edu.au
[2] Nanjing University of Science and Technology, Nanjing, China
longyuan@njust.edu.cn
[3] Shanghai Jiao Tong University, Shanghai, China

Abstract. Hop-constrained *s-t* simple path (HC-s-t path) enumeration is a fundamental problem in graph analysis. Existing solutions for this problem focus on unlabelled graphs and assume queries are issued without any label constraints. However, in many real-world applications, graphs are edge-labelled and the queries involve label constraints on the path connecting two vertices. Therefore, we study the problem of labelled hop-constrained s-t path (LHC-s-t path) enumeration in this paper. We aim to efficiently enumerate the HC-s-t paths using only edges with provided labels. To achieve this goal, we first demonstrate the existence of unnecessary computation specific to the label constraints in the state-of-the-art HC-s-t path enumeration algorithm. We then devise a novel online index to identify the fruitless exploration during the enumeration. Based on the proposed index, we design an efficient LHC-s-t path enumeration algorithm in which unnecessary computation is effectively pruned. Extensive experiments are conducted on real-world graphs with billions of edges. Experiment results show that our proposed algorithms significantly outperform the baseline methods by over one order of magnitude.

Keywords: Graph · Labelled graph · Path enumeration

1 Introduction

Graphs have been widely used to represent relationships of entities in many areas including social networks, web graphs, and biological networks [6,11,12,24]. With the proliferation of graph applications, research efforts have been devoted to many fundamental problems in analyzing graph [18,21–23]. Among them, the problem of hop-constrained s-t simple path (HC-s-t path) enumeration receives considerable attention and has been well-studied in the literature [3,5,15–17,19]. Given an unweighted directed graph G, a source vertex s, a target vertex t, and a hop constraint k, HC-s-t path enumeration computes all the simple paths (i.e.,

R. Chbeir et al. (Eds.): WISE 2022, LNCS 13724, pp. 49–64, 2022.
https://doi.org/10.1007/978-3-031-20891-1_5

paths without repeated vertices) from s to t such that the number of hops in each path is not larger than k.

Remarkably, all of the existing algorithms for HC-s-t path enumeration focus on unlabelled graphs and assume queries are issued without any label constraints. However, in many real-world applications, graphs are edge-labelled graphs [1,4,8,14,26], i.e., edges are associated with labels to denote different types of relationships between vertices, and the HC-s-t path enumeration for labelled graphs often involves label constraints on the path connecting two vertices. As a result, we study the problem of labelled hop-constrained s-t path (LHC-s-t path) enumeration in this paper. Specifically, given a labelled directed graph G, a source vertex s, a destination vertex t, a hop constraint k and a subset L of the set of all edge labels $\mathcal{L}$ of G, we aim to efficiently enumerate the HC-s-t paths using only edges with labels in L.

Applications. LHC-s-t path enumeration can be used in many applications, for example:

- *Fraud detection in E-commerce transaction networks.* A cycle in a E-commerce transaction network is a strong indication of a fraudulent activity [25]. A recent study from Alibaba group presents that the HC-s-t path enumeration is used to report all newly formed cycles to detect fraudulent activities [16]. In real world scenarios, transactions are usually associated with labels demonstrating their types, such as "bank_transfer" or "credit_card", and users may be interested in querying specific types of transactions (e.g. limiting the edge label to "credit_card" in credit card fraud detection). This gives rise to the need for LHC-s-t path enumeration.
- *Pathway queries in biological networks.* Pathway queries are a fundamental tool in biological networks analytics [7,10]. [10] shows that HC-s-t path enumeration is one of the most important pathway queries that can figure out the chains of interactions between two substances. Since there are many different types of interactions between two substances, the pathway queries are often issued with certain constraints on the types of interactions along the path, suggesting the utility of label constraints.
- *Path ranking in knowledge graphs.* Path ranking algorithms enumerate the paths from one entity to another in a knowledge graph and use these paths as features to train a model for missing fact prediction [2,9,13]. As real-world knowledge graphs (e.g., RDF graphs) are usually associated with rich labels, LHC-s-t path enumeration is used to enumerate paths with meaningful labels defined by users to more precisely predict the missing relations between entities.

Motivation and Challenges. To address this important problem, the most straightforward way is to directly process a LHC-s-t path query using the state-of-the-art HC-s-t path enumeration algorithm in the literature [19] (i.e. referred to as BaseEnum) and then remove the invalid results that violate the label constraints. However, although BaseEnum works well on general HC-s-t path queries, it is inefficient and unscalable to solve the problem of LHC-s-t path enumeration

especially on large-scale graphs. The reasons are as follows: (1) since BaseEnum is designed for general HC-s-t path queries, it does not consider the possible pruning opportunities to reduce unnecessary computation specific to the label constraints, while pruning such unnecessary computation has crucial effects on the enumeration performance as proved in our experiments; (2) as the results of BaseEnum may violate the label constraints, each HC-s-t path found by BaseEnum needs to be verified individually to ensure that the labels of edges in the path are within the given set of labels, which leads to a high verification cost.

Motivated by this, we aim to develop an efficient algorithm tailored for the problem of LHC-s-t path enumeration in large graphs. The algorithm should not only have the ability of pruning unnecessary computation specific to the label constraints during the enumeration, but also avoid the costly verification of the results at the end of the enumeration.

Contributions. The main contributions of our paper are summarized as follows:

(A) The first work to study the LHC-s-t path *enumeration.* To the best of our knowledge, this is the first work to study the problem of LHC-s-t path enumeration in large graphs, which has many real-world applications.

(B) An efficient and scalable algorithm for LHC-s-t path *enumeration.* By revisiting the missing pruning opportunities in the state-of-the-art HC-s-t path enumeration algorithm, we first propose an online label-based index based on which the fruitless exploration can be effectively identified. After that, we propose an efficient LHC-s-t path enumeration algorithm in which the identified unnecessary computation is effectively pruned, which can significantly improve the enumeration performance by reducing the search space.

(C) Extensive performance studies on real-world datasets. We conduct extensive performance studies on real-world graphs with various graph properties. The experiment results demonstrate that our proposed algorithm is efficient and scalable regarding processing HC-s-t path enumeration queries in large graphs.

2 Preliminary

Let $G = (V, E, \mathcal{L}, \lambda)$ denote a labelled directed graph, where $V(G)$ is the set of vertices, $E(G)$ is a set of directed edges, $\mathcal{L}(G)$ is the set of edge labels, and λ is the function that assigns each edge $e \in E(G)$ a label $\lambda(e) \in \mathcal{L}(G)$. We use n and m to denote the number of vertices and edges, respectively. For a vertex $v \in V(G)$, we use $G.\mathsf{nbr}^-(v)/G.\mathsf{nbr}^+(v)$ to denote the in-neighbors/out-neighbors of v in G. We omit G in the notations when the context is self-evident. Given a graph G, the reverse graph of G, denoted by $G_r = (V, E_r)$, is the graph generated by reversing the direction of all edges in G.

A path from vertex u to vertex v, denoted by $p(u, v)$, is a sequence of vertices $\{u = v_0, v_1, \ldots, v_h = v\}$ such that $(v_{i-1}, v_i) \in E(G)$ for every $1 \leq i < h$. Given two paths p_A and p_B, p_A is a partial path of p_B if p_A makes up part of P_B, denoted by $p_A \subseteq p_B$. A simple path is a loop-free path where there are no repetitions of vertices and edges. By $|p|$ and $p[i]$, we denote the length of path

p and the ith vertex of p, respectively. We call the intermediate path of a query during its enumeration as its prefix. Given two vertices u and v, the shortest distance from u to v, denoted by $\mathsf{dist}_G(u, v)$, is the length (i.e., the number of hops in this paper) of the shortest path from u to v on G, we omit G in the notations when the context is self-evident. Given a pre-defined hop constraint k, we say a path p is a hop-constrained path if $|p| \leq k$. We call a traversal on G as a forward search and on G_r as a backward search. The results of a LHC-s-t path query q are denoted as $P(q)$.

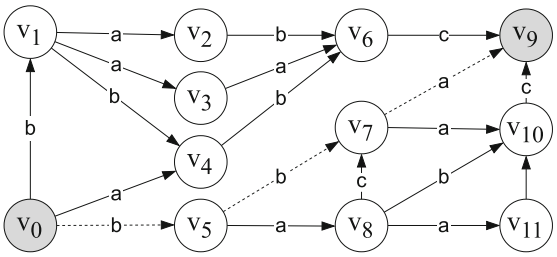


Fig. 1. Labelled graph G

Problem Statement. Given a labelled directed graph G and a LHC-s-t path query $q(s, t, k, L)$, we aim to *efficiently* enumerate the LHC-s-t path $P(q)$ from s to t on G using only edges with labels in L such that the number of hops in each path is not larger than k.

Example 1. Consider a labelled directed graph G shown in Fig. 1. Given a LHC-s-t path query $q(v_0, v_9, 4, \{a, b\})$, one LHC-s-t path can be found, namely $p = (v_0, v_5, v_7, v_9)$, which is demonstrated by the dashed arrows in Fig. 1.

3 The Baseline Solution

In this section, we introduce the state-of-the-art algorithm, PathEnum [19], for HC-s-t path enumeration query and present a straightforward baseline solution BaseEnum for LHC-s-t path query processing based on PathEnum. Given a HC-s-t path query q from s to t with hop constraint k, the main idea of PathEnum is to prune any vertex v visited during the enumeration if the shortest distance $\mathsf{dist}(v, t)$ from v to t exceeds the remaining hop budget.

Based on PathEnum, the baseline solution for LHC-s-t path query processing is as follows: since the results computed by PathEnum contain all the results for LHC-s-t path enumeration, it is immediate that all LHC-s-t paths for q can be correctly enumerated by filtering the HC-s-t paths that violate the label constraints. Algorithm 1 illustrates the baseline solution BaseEnum for LHC-s-t path query processing.

Algorithm 1: BaseEnum(G, q)

```
1  Find dist_{G_r}(q.t, v) for each v ∈ V(G) by a BFS;
2  Search(G, P, (q.s), q);
3  P ← Verify(P, q);
4  foreach p ∈ P do
5  |  Output p;
6  Procedure Search(G, P, p, q)
7  |  v' ← p[|p|]; if v' = q.t then P.add(p);
8  |  if |p| = q.k or v' = q.t then return;
9  |  foreach v'' ∈ nbr+(v') s.t. |p| + dist_{G_r}(q.t, v'') < k do
10 |  |  if v'' ∉ p then Search(G, P, p⋃{v''}, q);
11 Procedure Verify(P, q)
12 |  P' ← ∅;
13 |  foreach p ∈ P do
14 |  |  foreach i ∈ 0..|p| − 1 do
15 |  |  |  if λ((p[i], p[i + 1])) ∉ q.L then
16 |  |  |  |  continue;
17 |  |  |  P'.add(p);
18 |  return P';
```

Given a graph G and a LHC-s-t path query q, BaseEnum first computes the distance between t and all vertices in G by running a BFS (line 1). Then the enumeration for LHC-s-t paths is conducted (line 2). The paths obtained by the search are then verified to ensure that only paths with valid labels are output (lines 3–5). Procedure Search enumerates the paths with a hop constraint of k recursively (lines 6–10). Specifically, if a path p ends with vertex $q.\mathsf{t}$ is found, p is a candidate result, which is immediately added to P (line 8). If p's length has reached $q.\mathsf{k}$ or p ends with vertex $q.\mathsf{t}$, the search terminates (line 9). Otherwise, for the out-neighbor v'' of v' which meets the hop constraint and has not been explored in p, BaseEnum adds v'' in p and continues the search (line 9–10). In procedure Verify, the paths found by Search are examined in a row, such that any path that contains invalid labels are filtered and the rest are returned as the final results for q (lines 11–18).

Remark. As a bidirectional search could improve the enumeration performance, PathEnum conducts a forward search from s on G and a backward search from t on G_r concurrently, and concatenates the paths explored during the bidirectional search to obtain the final results by a hash join. Since the optimization is orthogonal to the label constraints in our problem and can be easily adapted to our approach, we follow the single direction search when introducing BaseEnum and our approach in Sect. 4. In Sect. 7, both of these methods are evaluated in our experiments.

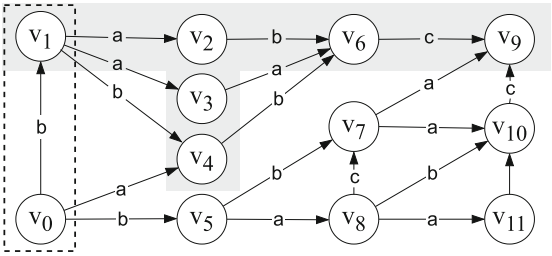


Fig. 2. Main observation of our approach

4 Our Approach

4.1 Overview

Given a LHC-s-t path query, BaseEnum prunes the unnecessary computation during enumeration by examining the vertices' reachability based on the existence of edges, which misses the possible pruning opportunities specific to the label constraints. In fact, there still exists a huge amount of fruitless exploration attributed to the label constraints, which can be identified and avoided to significantly reduce the search space. Consider the enumeration procedure of BaseEnum regarding $q(v_0, v_9, 4, \{a, b\})$ in Fig. 1. Initially, the search by BaseEnum starts from v_0. Because $|(v_0)| + \mathsf{dist}(v_1, v_9) = 4 \leq q.\mathsf{k}$, the search following v_1 is determined to be promising and hence v_1 is explored, resulting in the search prefix (v_0, v_1) shown in Fig. 2 by the dashed box. After this, because it is found by looking up the index that for each $v \in \{v_2, v_3, v_4\}$, $|(v_0, v_1)| + \mathsf{dist}(v, v_9) = 4 \leq q.\mathsf{k}$, all three out-neighbors of v_1 are considered to be promising and therefore explored, resulting in the enumerated prefixes $\{(v_0, v_1, v_2), (v_0, v_1, v_3), (v_0, v_1, v_4)\}$. Subsequently, v_6 and v_9 are explored in a row. However, when verifying the correctness of the three found paths $\{(v_0, v_1, v_2, v_6, v_9), (v_0, v_1, v_3, v_6, v_9), (v_0, v_1, v_4, v_6, v_9)\}$, it is noticed that all these results are invalid because $\lambda((v_6, v_9)) = c$ and thus the label constraint $q.\mathsf{L} = \{a, b\}$ fails on edge (v_6, v_9) in all of these paths. As a result, all the computation in this example is fruitless and therefore unnecessary, which is represented by the shaded area in Fig. 2.

According to this example, it is clear that there exists a huge amount of unnecessary computation specific to label constraints in LHC-s-t path enumeration, while pruning them in advance can considerably reduce the search space and avoid the expensive final verification, which consequently improves the enumeration performance. To achieve this goal, we first define a label-based index, which is constructed online to effectively identify the fruitless exploration. Based on the devised index, we propose an efficient LHC-s-t path enumeration algorithm which can maximally prune the unnecessary computation, and thus improve the whole performance.

4.2 Label-Based Index

As discussed in Sect. 4.1, given a LHC-s-t path query q, pruning the exploration that violates the label constraints can significantly reduce the search space. However, it is infeasible to prune such exploration by dynamically checking each explored edge during the enumeration to ensure that its label is in $q.\mathsf{L}$. Consider the same example in Fig. 2, where the prefix (v_0, v_1) explores v_2, v_3 and v_4 by extending the edges (v_1, v_2), (v_1, v_3) and (v_1, v_4). Because $\lambda((v_1, v_2)) = a$, $\lambda((v_1, v_3)) = a$ and $\lambda((v_1, v_4)) = b$, all these three edges have valid labels contained by $q.\mathsf{L}$, which means the search on them should continue normally. However, it has been demonstrated in Sect. 4.1 that the enumeration following them is fruitless. According to this contradiction, it is obvious that the fruitless computation attributed to the label constraints cannot be trivially identified and pruned by dynamically checking the labels of explored edges. As a result, the following question arises: is it possible to design an approach such that the fruitless exploration regarding LHC-s-t path enumeration can be effectively identified and pruned? In this section, we aim to answer this questions and introduce our approach to prune the unpromising search in LHC-s-t path enumeration.

Reconsider the example of running $q(v_0, v_9, 4, \{a, b\})$ on G by BaseEnum shown in Fig. 2, we observe that the enumeration following (v_0, v_1) is fruitless because v_1 is not reachable to v_9 due to the label constraint, which only allows edges labelled a and b to be explored during the search. Based on this observation, if the reachability regarding the query constraints on both labels and hops can be efficiently retrieved, the pruning can be effectively done during the enumeration. Following the idea, we define:

Definition 1 ***(Label-constrained Distance*** $\mathscr{D}$**).** *Given a graph G, two vertices v_0, v_1 and a set of labels L, the label-constrained distance from v_0 to v_1 with labels L, denoted by $\mathscr{D}_G(v_0, v_1, L)$, is the length of the shortest path from v_0 to v_1 on G using only the labels in L. G is omitted in the notation when the context is self-evident.*

Based on the definition of label-constrained distance, we then prove the following lemmas on which our index is based:

Lemma 1. *Given a graph G, a* LHC-s-t path *query q and a vertex $v \in V(G)$, if there exists a* LHC-s-t path *p from s to t with $|p| \leq k$ and constrained label set L, then for any $p[i] = v$ where $0 \leq i < |p|$, $\mathscr{D}_G(s, v, L) \leq i$ and $\mathscr{D}_{G_r}(t, v, L) \leq k - i$.*

Proof. Based on the definition of LHC-s-t path, if $0 \leq i < |p|$, then $v[i]$ must be reachable from $v[0]$ (i.e. s) and reachable to $v[|p|]$ (i.e. t) within i and $|p| - i$ hops, respectively, using only the labels in $q.\mathsf{L}$. Moreover, given that $|p| \leq k$ and $v[i]$ must be reachable to $v[|p|]$ (i.e. t) within $k - i$ hops using only the labels in $q.\mathsf{L}$, $v[i]$ should also be reachable to t within $k - i$ hops regarding $q.\mathsf{L}$. □

According to Lemma 1, we further define the following lemmas to demonstrate how the vertices can be pruned during the search to reduce unnecessary computation:

Algorithm 2: ConstructIndex(G, q)

```
1  S_v ← {q.s}; end ← False;
2  D ← ∅; D(q.s, q.s, q.L) = 0;
3  while |S_v| ≠ 0 and !end do
4  |  v ← S_v[0]; S_v.remove(v);
5  |  foreach v' ∈ nbr+_G(v) s.t. λ((q.s, v')) ∈ q.L do
6  |  |  if (q.s, v', q.L) ∉ D then
7  |  |  |  if q.k < D(q.s, v, q.L) + 1 then
8  |  |  |  |  end = True; break;
9  |  |  |  D(q.s, v', q.L) = D(q.s, v, q.L) + 1;
10 |  |  |  S_v.add(v');
11 return D;
```

Lemma 2. *Given a graph G, a vertex $v \in V(G)$ and a* LHC-s-t path *query q, if $\mathcal{D}_G(q.\mathsf{s}, v, q.\mathsf{L}) + \mathcal{D}_{G_r}(q.\mathsf{t}, v, q.\mathsf{L}) > q.k$, then $\nexists p \in P(q)$ s.t. $v \in p$.*

Proof. Based on Lemma 1, if $\mathcal{D}_G(q.\mathsf{s}, v, q.\mathsf{L}) + \mathcal{D}_{G_r}(q.\mathsf{t}, v, q.\mathsf{L}) > q.k$, then when v exists in a LHC-s-t path p from s to t, the length $|p|$ of p must be larger than $q.\mathsf{k}$, thus p obviously cannot be a valid result for q, which proves $\nexists p \in P(q)$ s.t. $v \in p$. □

According to Lemma 2, given a LHC-s-t path query q, it is clear that for any vertex $v \in V(G)$, if $\mathcal{D}_G(q.\mathsf{s}, v, q.\mathsf{L}) + \mathcal{D}_{G_r}(q.\mathsf{t}, v, q.\mathsf{L}) > q.k$, then vertex v can never appear in a valid LHC-s-t path and thus does not need to be explored at all during the enumeration, which can be removed from $V(G)$ before the enumeration begins. Moreover, the pruning can also take place during the enumeration, as shown in the following lemma:

Lemma 3. *Given a graph G, a prefix p' and a* LHC-s-t path *query q, if $|p'| + \mathcal{D}_{G_r}(q.\mathsf{t}, p'[|p'|], q.\mathsf{L}) > q.k$, then $\nexists p \in P(q)$ s.t. $p' \subseteq p$.*

Proof. Based on Lemma 1, if $|p'| + \mathcal{D}_{G_r}(q.\mathsf{t}, p'[|p'|], q.\mathsf{L}) > q.k$, then p' requires at least $q.k - |p'| + 1$ hops to explore $q.\mathsf{t}$, which obviously exceeds the hop budget left. As a result, $\nexists p \in P(q)$ s.t. $p' \subseteq p$. □

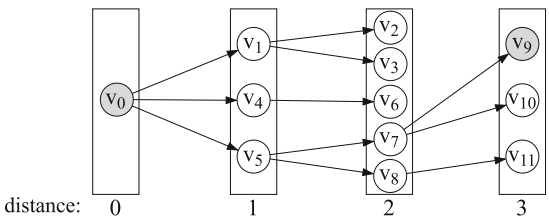
According to Lemma 3, given a LHC-s-t path query q and a prefix p', if $|p'| + \mathcal{D}_{G_r}(q.\mathsf{t}, p'[|p'|], q.\mathsf{L}) > q.k$, then p' requires to proceed more hops to reach $q.\mathsf{t}$ than its remaining hop budget. Hence the enumeration procedure following p' is fruitless and can be directly pruned.

Based on the observations, given a graph G and a LHC-s-t path query q, in order to reduce unnecessary computation during the enumeration, we want to build an index to support instant lookup on both $\mathcal{D}_G(q.\mathsf{s}, v, q.\mathsf{L})$ and $\mathcal{D}_{G_r}(q.\mathsf{t}, v, q.\mathsf{L})$ for $v \in V(G)$, which gives rise to our index construction algorithm ConstructIndex.

Algorithm. Algorithm 2 illustrates the procedure of the index construction on G, as the same procedure on G_r can be achieved similarly. Given a graph G and

a LHC-s-t path query q, ConstructIndex performs a BFS exploration on G while updating the label-constrained distance $\mathscr{D}$ from $q.\mathsf{s}$ to each visited vertex.

Specifically, ConstructIndex first creates a queue S_v to store the current set of vertices whose out-neighbors need to be explored, which initially only contains $q.\mathsf{s}$. Boolean variable *end* is created to record if the index construction needs to terminate, which happens when the allowed hop budget has been exhausted (line 1). Then, $\mathscr{D}$ is initialized and the label-constrained distance between $q.\mathsf{s}$ and itself is set to 0 (line 2). After this, ConstructIndex iteratively explores the vertices while following the label constraints in G in a BFS order, updating the label-constrained distance $\mathscr{D}$ for them, which terminates when no more vertices can be explored or the distance has exceeded the hop constraint (lines 3–10). At the start, the first vertex v in S_v is popped out and v's out-neighbors v' such that the label of edge (v, v') is in $q.\mathsf{L}$ are visited (lines 4–5). When the distance between $q.\mathsf{s}$ and v' is not recorded, if the new distance to be assigned, $\mathscr{D}(q.\mathsf{s}, v, q.\mathsf{L})+1$, is greater than $q.\mathsf{k}$, then the assigned distance has exceeded the hop constraint. Therefore, the procedure of ConstructIndex will terminate (lines 6–8). Otherwise, $\mathscr{D}(q.\mathsf{s}, v', q.\mathsf{L})$ is updated and v' is added to S_v for further exploration (lines 9–10). Finally, $\mathscr{D}$ is returned (line 11).



(a) Graph Exploration

v	$\mathscr{D}(v_0, v, L)$	v	$\mathscr{D}(v_0, v, L)$
v_0	0	v_6	2
v_1	1	v_7	2
v_4	1	v_8	2
v_5	1	v_9	3
v_2	2	v_{10}	3
v_3	2	v_{11}	3

(b) Index

Fig. 3. Example of ConstructIndex

Example 2. Reconsider the graph G and the LHC-s-t path query $q(v_0, v_9, 4, \{a, b\})$ in Fig. 1. We demonstrate the example of running Algorithm 2 on (G, q) in detail, where the vertices explored at each size of the label constraint distance are shown in Fig. 3(a). The vertices $q.\mathsf{s}$ and $q.\mathsf{t}$ are marked in grey. Additionally, Fig. 3(b) demonstrates the index constructed on $\mathscr{D}$, which shows the label-constrained distance between s (i.e. v_0) and each $v \in V(G)$ that is reachable from s within the hop budget.

Initially, the search starts from v_0 because $S_v = \{v_0\}$. Then, the out-neighbors of v_0, namely $\{v_1, v_4, v_5\}$, are visited as they follow the label constraint, resulting in the new label-constraint distances $\mathscr{D}(v_0, v_1, q.\mathsf{L}) = \mathscr{D}(v_0, v_4, q.\mathsf{L}) = \mathscr{D}(v_0, v_5, q.\mathsf{L}) = 1$, which are inserted into the index. After this, the vertices in S_v, namely $\{v_1, v_4, v_5\}$, are extended by their out-neighbors $\{v_2, v_3, v_6, v_7, v_8\}$ and their label-constraint distances from v_0 are also updated to 2. Finally, because the

Algorithm 3: LabelledEnum(G, q)

```
1  D ← ConstructIndex(G/G_r, q);
2  ReduceGraph(G, q, D); P ← ∅;
3  Search(G, P, (q.s), q);
4  foreach p ∈ P do
5  |  Output p;
6  Procedure ReduceGraph(G, q, D)
7  |  foreach v ∈ V(G) do
8  |  |  if D_G(q.s, v, q.L) + D_{G_r}(q.t, v, q.L) > q.k then
9  |  |  |  V(G).remove(v);
10 |  |  |  foreach e ∈ E(G) s.t. v ∈ e do
11 |  |  |  |  E(G).remove(e);
12 Procedure Search(G, P, p, q)
13 |  v' ← p[|p|]; if v' = q.t then P.add(p);
14 |  if |p| = q.k or v' = q.t then return;
15 |  foreach v'' ∈ nbr+(v') s.t. |p| + D_{G_r}(q.t, v') < k do
16 |  |  if v'' ∉ p then Search(G, P, p ⋃ {v''}, q);
```

label of edge (v_6, v_9) is c which violates the label constraint, vertex v_9 cannot be extended from v_6. Instead, v_9 and v_{10} are extended from v_7, and v_{11} is extended from v_8, leading to $\mathscr{D}(v_0, v_9, q.\mathsf{L}) = \mathscr{D}(v_0, v_{10}, q.\mathsf{L}) = \mathscr{D}(v_0, v_{11}, q.\mathsf{L}) = 3$.

Theorem 1. *The time complexity of* ConstructIndex *is* $O(|V(G)| + |E(G)|)$.

Proof. The time complexity of ConstructIndex is $O(|V(G)| + |E(G)|)$ because a breadth-first search is performed during the construction procedure, which visits each vertex and edge in G once. □

4.3 LHC-s-t path Enumeration

Given a graph G and a LHC-s-t path query q, after constructing the index on the label-constrained distances from $q.\mathsf{s}/q.\mathsf{t}$ to each $v \in V(G)$ (the label-constrained distance to vertices that are not visited in ConstructIndex are treated as ∞.), we can identify the fruitless exploration during the enumeration based on Lemma 2 and Lemma 3, and thus accelerate the LHC-s-t path query processing by pruning them. According to Lemma 2, it is obvious that there exist many vertices in $V(G)$ that will never be a part of a valid LHC-s-t path due to the label and hop constraints. As a result, to minimize the unnecessary computation, we remove such vertices in $V(G)$ and their corresponding edges in $E(G)$ in advance, which effectively reduces the graph size and search space before the enumeration procedure begins. Additionally, based on Lemma 3, it can be determined early that the vertices explored during the enumeration will be fruitless. Therefore, we dynamically verify the explored vertices to ensure that the search following them must be promising. The detailed algorithm, LabelledEnum, is shown in Algorithm 3.

Algorithm. For a LHC-s-t path query q, LabelledEnum first constructs the index on both G and G_r, to compute $\mathscr{D}_G(q.\mathsf{s}, v)$ and $\mathscr{D}_{G_r}(q.\mathsf{t}, v)$ for all $v \in V(G)$ (line

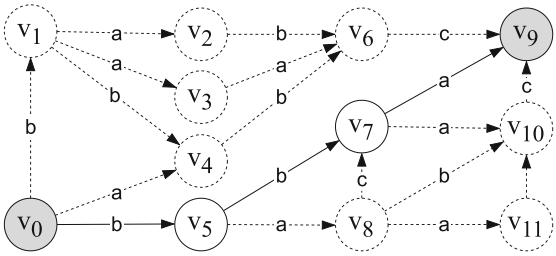


Fig. 4. Example of ConstructIndex

1). After this, LabelledEnum reduces the search space by removing the vertices and edges in G that will never be a part of a valid LHC-s-t path (line 2). P is then initialized to store the LHC-s-t paths found during the enumeration, which is conducted in a DFS manner with procedure search (line 3). For each path p obtained during the search, p is output as a valid LHC-s-t path (lines 4–5). In procedure ReduceGraph, each vertex $v \in V(G)$ such that $\mathscr{D}_G(q.\mathsf{s}, v, q.\mathsf{L}) + \mathscr{D}_{G_r}(q.\mathsf{t}, v, q.\mathsf{L}) > q.k$ will be removed directly based on Lemma 2 (lines 7–9). Any edge that contains such v is also removed (lines 10- 11). Procedure Search enumerates the LHC-s-t paths with a hop constraint of k based on the index recursively (lines 12–16). Specifically, if a path p with length $q.\mathsf{k}$ is found, p is added into P (line 14). Otherwise, for the out-neighbor v'' of v' which meets both the label and hop constraint according to Lemma 3 and has not been explored in p, LabelledEnum adds v'' in p and continues the search (line 15–16).

Example 3. Reconsider the graph G and LHC-s-t path query q in Fig. 1. We demonstrate the example of running LabelledEnum on G and q. After constructing the index, the dashed vertices and edges in Fig. 4 are those removed in procedure ReduceGraph due to the violation of label and hop constraints. The search starts from prefix (v_0) and explores v_5 because $|(v_0)| + \mathscr{D}_{G_r}(v_9, v_5) = 2 < q.\mathsf{k}$. Similarly, v_7 and v_9 are subsequently explored, resulting in the found LHC-s-t path (v_0, v_5, v_7, v_9).

Theorem 2. *Given a graph G and a* LHC-s-t path *query q, Algorithm 3 enumerates all* LHC-s-t paths *for q in G correctly.*

Proof. Based on the correctness of PathEnum, it is direct that q's LHC-s-t paths on G are enumerated correctly if no pruning is done on the label constraints. Moreover, according to Lemma 2 and Lemma 3, a vertex is not pruned unless it cannot be a part of a valid LHC-s-t path. As a result, Algorithm 3 enumerates all LHC-s-t paths for q in G correctly. □

5 Evaluation

In this section, we evaluate the efficiency of the proposed algorithms. All the experiments are performed on a machine with one 20-core Intel Xeon CPU E5-2698 and 512 GB main memory running Red Hat Linux 7.3, 64 bit.

Table 1. Statistics of the datasets

Dataset	Name	$\|V\|$	$\|E\|$	d_{avg}	d_{max}
Epinsion	EP	75K	508K	13.4	3,079
LiveJournal	LJ	4M	69M	17.9	20,333
Twitter-2010	TW	42M	1.46B	70.5	2,997,487
Friendster	FS	65M	1.81B	27.5	5,214

Datasets. We evaluate our algorithms on four real-world graphs, which are shown in Table 1. Among them, Epinsion is a who-trust-whom online social network of consumer review site Epinions.com. LiveJournal is a free online community based social network. Twitter-2010 is a web graph crawled from Twitter. Friendster is a social network retrieved from Friendster. Twitter-2010 is downloaded from LAW[1] and the rest are downloaded from SNAP[2]. As all the datasets come with no edge labels, we use the method in [20] to synthetize the edge labels, while the number of labels $|\mathcal{L}(G)|$ is set to 8.

Algorithms. We compare the following algorithms:

- BaseEnum: Algorithm 1 where PathEnum runs in a single direction (Sect. 3).
- BaseEnum$^+$: Algorithm 1 where PathEnum runs with an optimized bidirectional search order (Sect. 3).
- LabelledEnum: Algorithm 3 (Sect. 4.3).
- LabelledEnum$^+$: LabelledEnum with an optimized bidirectional search order introduced by BaseEnum$^+$ (Sect. 3).

Settings. All the algorithms are implemented in Rust 1.43. In the experiments, the time cost is measured as the amount of wall-clock time elapsed during the program's execution.

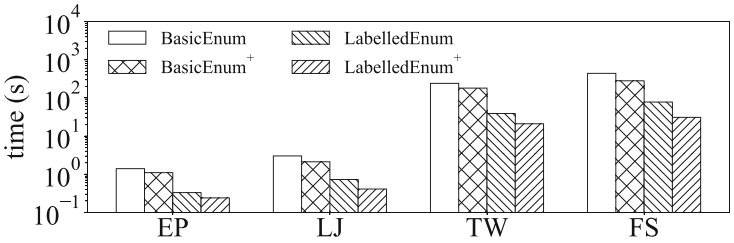
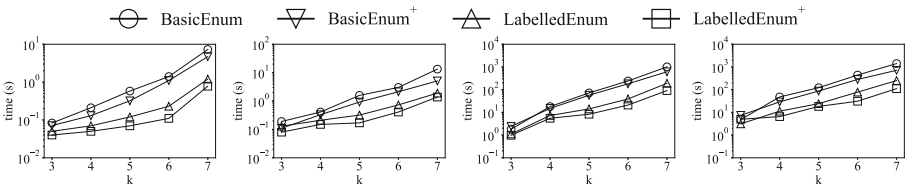


Fig. 5. Processing time on all datasets

[1] https://law.di.unimi.it/index.php.
[2] https://snap.stanford.edu/data/.



(a) EP (b) LJ (c) TW (d) FS

Fig. 6. Processing time when varying k

Exp-1: Efficiency on Different Datasets. In this experiment, we evaluate the processing time of four algorithms (i.e. BaseEnum, BaseEnum$^+$, LabelledEnum, LabelledEnum$^+$) on all four graphs. We randomly generate 100 random LHC-s-t path query $q(s, t, k, L)$ on each graph where source vertex s could reach target vertex t in k hops with the given label set L. We set $k = 6$ and $|L| = 4$ by default. The results are reported in Fig. 5.

As can be seen in Fig. 5, our proposed LHC-s-t path enumeration algorithms LabelledEnum and LabelledEnum$^+$ always outperform the other two algorithms on all of the graphs. For example, on graph FS, LabelledEnum is 5.6× faster than BaseEnum and 3.5× faster than BaseEnum$^+$. Comparatively, LabelledEnum$^+$ demonstrates a better performance, which is 12.9× faster than BaseEnum and 8.1× faster than BaseEnum$^+$. This is because the pruning technique used in our proposed algorithms can significantly reduce the search space and the fruitless exploration attributed to the label constraints is avoided. For LabelledEnum and LabelledEnum$^+$, LabelledEnum$^+$ always outperforms LabelledEnum on all datasets, this is because by using the bidirectional search, some computed paths can be shared during the enumeration, which is consistent with the analysis in [19].

Exp-2: Efficiency When Varying Hop Constraint k. In this experiment, we evaluate the efficiency when varying hop constraint k from 3 to 7. For each hop constraint k, we randomly generate 50 queries. The average processing time for each query is shown in Fig. 6.

As shown in Fig. 6, as the hop constraint k increases, the processing time of all algorithms increases as well. This is because as k increases, the number of LHC-s-t paths also increases. Furthermore, LabelledEnum and LabelledEnum$^+$ always outperform the other two algorithms, and the performance gap increases as the hop constraint k increases. For example on TW, when $k = 4$, LabelledEnum is 2.4× faster than BaseEnum and 2.1× faster than BaseEnum$^+$; in contrast, when k increases to 7, LabelledEnum becomes 5.6× faster then BaseEnum and 3.5× faster than BaseEnum$^+$. This is because when the search space grows larger due to the increase of k, the fruitless exploration in the baseline algorithms increases accordingly while these fruitless exploration can be significantly avoided due to index-based pruning on the label constraints in LabelledEnum and LabelledEnum$^+$.

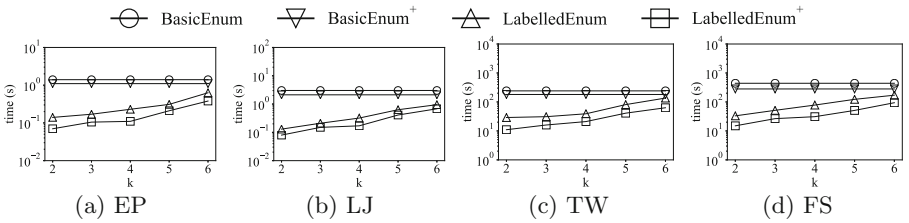


Fig. 7. Processing time when varying $|L|$

Exp-3: Efficiency When Varying Label Set Size $|L|$. In this experiment, we evaluate the efficiency when varying the label set size $|L|$ from 2 to 6. For each label set size $|L|$, we randomly generate 50 queries. The average processing time for each query is shown in Fig. 7.

As shown in Fig. 7, as the label set size $|L|$ increases, the processing time of our proposed algorithms increases while the time of the two baseline algorithms remain the same. This is because as $|L|$ increases, the size of fruitless computation that can be pruned becomes smaller. For example on TW, when $|L| = 2$, LabelledEnum is 8.3× faster than BaseEnum and 6.3× faster than BaseEnum$^+$; in contrast, when $|L|$ increases to 6, LabelledEnum becomes 1.8× faster then BaseEnum and 1.4× faster than BaseEnum$^+$. The time of the two baseline algorithms remain the same because they always have to first enumerate all the paths that follow the hop constraint.

6 Related Work

HC-s-t path enumeration is a fundamental problem in graph analysis and several algorithms have been proposed [3,15–17,19] for this problem. We divided them into two categories, which are *pruning-based algorithms* [3,15,17] and *index-based algorithms* [19]. Pruning-based algorithms typically adopt a backtracking strategy based on a depth-first search based framework. During the enumeration, [17] and [3] dynamically compute the shortest path distance from v to t and prunes v if it is unreachable to t, while [15] dynamically maintains a lower bound of hops to the target vertex t for the vertices visited and prunes v if the current remaining hop budget is smaller than the lower bound of hops required. For the index-based algorithm, [19] finds that the pruning-based algorithms typically suffer from severe performance issues caused by the costly pruning operations during enumeration. Therefore, BaseEnum builds a light-weight index to reduce the number of edges involved in the enumeration. Thanks to the index structure, BaseEnum significantly outperform pruning-based algorithms as demonstrated in [19]. Hence, we also adopt the index-based approach in this paper.

7 Conclusion

In this paper, we study the problem of LHC-s-t path enumeration. To address this problem, we observe that there exists a huge amount of unnecessary computation due to the label constraints and the processing performance can be significantly improved if these unnecessary computation can be effectively pruned. Following this observation, we first propose a label-based index to identify the fruitless exploration. Based on the constructed index, we design an efficient algorithm to process the given query by pruning the fruitless computation related to both the label and hop constraints. Experiment results on real-world datasets demonstrate the efficiency of our proposed algorithms.

References

1. Chen, Z., Yuan, L., Lin, X., Qin, L., Yang, J.: Efficient maximal balanced clique enumeration in signed networks. In: WWW 2020: The Web Conference 2020, Taipei, Taiwan, 20–24 April 2020, pp. 339–349. ACM/IW3C2 (2020)
2. Freitas, A., da Silva, J.C.P., Curry, E., Buitelaar, P.: A distributional semantics approach for selective reasoning on commonsense graph knowledge bases. In: Métais, E., Roche, M., Teisseire, M. (eds.) NLDB 2014. LNCS, vol. 8455, pp. 21–32. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07983-7_3
3. Grossi, R., Marino, A., Versari, L.: Efficient algorithms for listing k disjoint st-paths in graphs. In: Bender, M.A., Farach-Colton, M., Mosteiro, M.A. (eds.) LATIN 2018. LNCS, vol. 10807, pp. 544–557. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-77404-6_40
4. Hao, K., Yang, Z., Lai, L., Lai, Z., Jin, X., Lin, X.: PatMat: a distributed pattern matching engine with cypher. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, pp. 2921–2924 (2019)
5. Hao, K., Yuan, L., Zhang, W.: Distributed hop-constrained s-t simple path enumeration at billion scale. Proc. VLDB Endow. **15**(2), 169–182 (2021)
6. Hao, Y., Zhang, Y., Cao, J.: A novel QoS model and computation framework in web service selection. World Wide Web **15**(5), 663–684 (2012)
7. Krishnamurthy, L., et al.: Pathways database system: an integrated system for biological pathways. Bioinform. **19**(8), 930–937 (2003)
8. Lai, L., et al.: Distributed subgraph matching on timely dataflow. Proc. VLDB Endow. **12**(10), 1099–1112 (2019)
9. Lao, N., Cohen, W.W.: Relational retrieval using a combination of path-constrained random walks. Mach. Learn. **81**(1), 53–67 (2010)
10. Leser, U.: A query language for biological networks. In: ECCB/JBI 2005 Proceedings, Madrid, Spain, 28 September–1 October 2005, p. 39 (2005)
11. Li, L., Xu, G., Yang, Z., Dolog, P., Zhang, Y., Kitsuregawa, M.: An efficient approach to suggesting topically related web queries using hidden topic model. World Wide Web **16**(3), 273–297 (2013)
12. Liu, B., Yuan, L., Lin, X., Qin, L., Zhang, W., Zhou, J.: Efficient (a, β)-core computation: an index-based approach. In: The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, 13–17 May 2019, pp. 1130–1141. ACM (2019)
13. Mazumder, S., Liu, B.: Context-aware path ranking for knowledge base completion. In: Sierra, C. (ed.) IJCAI, pp. 1195–1201. ijcai.org (2017)

14. Peng, Y., Lin, X., Zhang, Y., Zhang, W., Qin, L.: Answering reachability and K-reach queries on large graphs with label-constraints. VLDB J. **31**, 1–25 (2021)
15. Peng, Y., Zhang, Y., Lin, X., Zhang, W., Qin, L., Zhou, J.: Hop-constrained s-t simple path enumeration: towards bridging theory and practice. Proc. VLDB Endow. **13**(4), 463–476 (2019)
16. Qiu, X., et al.: Real-time constrained cycle detection in large dynamic graphs. Proc. VLDB Endow. **11**(12), 1876–1888 (2018)
17. Rizzi, R., Sacomoto, G., Sagot, M.-F.: Efficiently listing bounded length *st*-paths. In: Kratochvíl, J., Miller, M., Froncek, D. (eds.) IWOCA 2014. LNCS, vol. 8986, pp. 318–329. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19315-1_28
18. Sun, R., Chen, C., Liu, X., Xu, S., Wang, X., Lin, X.: Critical nodes identification in large networks: the inclined and detached models. World Wide Web **25**(3), 1315–1341 (2022)
19. Sun, S., Chen, Y., He, B., Hooi, B.: PathEnum: towards real-time hop-constrained s-t path enumeration. In: Proceedings of SIGMOD, pp. 1758–1770 (2021)
20. Valstar, L.D.J., Fletcher, G.H.L., Yoshida, Y.: Landmark indexing for evaluation of label-constrained reachability queries. In: SIGMOD, pp. 345–358. ACM (2017)
21. Wang, K., Lin, X., Qin, L., Zhang, W., Zhang, Y.: Accelerated butterfly counting with vertex priority on bipartite graphs. VLDB J. 1–25 (2022)
22. Wang, K., Zhang, W., Lin, X., Qin, L., Zhou, A.: Efficient personalized maximum biclique search. In: 2022 IEEE 38th International Conference on Data Engineering (ICDE), pp. 498–511. IEEE (2022)
23. Yang, Z., Lai, L., Lin, X., Hao, K., Zhang, W.: HUGE: an efficient and scalable subgraph enumeration system. In: Li, G., Li, Z., Idreos, S., Srivastava, D. (eds.) SIGMOD 2021: International Conference on Management of Data, Virtual Event, China, 20–25 June 2021, pp. 2049–2062. ACM (2021)
24. Yao, W., He, J., Huang, G., Cao, J., Zhang, Y.: A graph-based model for context-aware recommendation using implicit feedback data. World wide web **18**(5), 1351–1371 (2015)
25. Yue, D., Wu, X., Wang, Y., Li, Y., Chu, C.H.: A review of data mining-based financial fraud detection research. In: 2007 International Conference on Wireless Communications, Networking and Mobile Computing, pp. 5519–5522. IEEE (2007)
26. Zhang, J., Yuan, L., Li, W., Qin, L., Zhang, Y.: Efficient label-constrained shortest path queries on road networks: a tree decomposition approach. Proc. VLDB Endow. **15**(3), 686–698 (2021)