# An Experimental Study on Hub Labeling based Shortest Path Algorithms

Ye Li[#1] Leong Hou U[#2] Man Lung Yiu[*3] Ngai Meng Kou[#4]

[#]Department of Computer and Information Science, University of Macau

[1]yb47438@umac.mo [2]ryanlhu@umac.mo [4]yb27406@umac.mo

[*]Department of Computing, Hong Kong Polytechnic University

[3]csmlyiu@comp.polyu.edu.hk

## ABSTRACT

Shortest path distance retrieval is a core component in many important applications. For a decade, hub labeling (HL) techniques have been considered as a practical solution with fast query response time (e.g., 1-3 orders of magnitude faster), competitive indexing time, and slightly larger storage overhead (e.g., several times larger). These techniques enhance query throughput up to hundred thousands queries per second, which is particularly helpful in large user environment. Despite the importance of HL techniques, we are not aware of any comprehensive experimental study on HL techniques. Thus it is difficult for a practitioner to adopt HL techniques for her applications.

To address the above issues, we provide a comprehensive experimental study on the state-of-the-art HL technique with analysis of their efficiency, effectiveness and applicability. From insightful summary of different HL techniques, we further develop a simple yet effective HL techniques called Significant path based Hub Pushing (SHP) which greatly improves indexing time of previous techniques while retains good query performance. We also complement extensive comparisons between HL techniques and other shortest path solutions to demonstrate robustness and efficiency of HL techniques.

**PVLDB Reference Format:**
Ye Li, Leong Hou U, Man Lung Yiu, Ngai Meng Kou. An Experimental Study on Hub Labeling based Shortest Path Algorithms. *PVLDB*, 11(4): 445-457, 2017.
DOI: https://doi.org/10.1145/3164135.3164141

## 1. INTRODUCTION

Shortest path distance retrieval is a core component in several application domains, e.g., route planning, location-based games, recommendation and analytics on social networks. The shortest path problem has received abundant attention over the past few decades. Dijkstra's algorithm [20] is the first study to find the shortest path by network traversal, which takes $O(|E| + |V| \log |V|)$ time. Many subsequent work have been developed based on the network traversal paradigm over the past five decades, e.g., A* [27], ALT [24], REACH [26], HH [34], CH [23], AH [40], TNR [10], REAL [25], ReachFlags [12], SHARC [11], CALT [12], and CHASE [12]. However, these solutions incur considerable time on traversing a network. In addition, these methods are specifically designed to exploit road network properties (e.g., highway dimension [4, 23, 34, 40]), rendering them ineffective for complex networks (e.g., social, communication, hyperlink, and citation networks).

Another line of solutions is to retrieve the shortest path distance without network traversal. Recent studies attempt to answer the shortest path distance rapidly by hub labeling (HL) techniques. The first HL algorithm was proposed by Cohen et al. a decade ago [15]. The main idea of *hub labeling* is to precompute a set of hub labels for each vertex. At query time, the shortest path distance from source $s$ to destination $t$ can be computed by a sort-merge join between the label sets of $s$ and $t$.

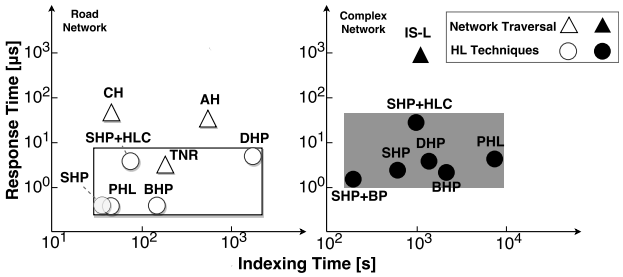
### 1.1 Motivations

To the best of our understanding, HL techniques are the only all-round solutions that can compute shortest path distance in microseconds for all types of large networks (e.g., with millions of vertices), which is particularly helpful in large user environment (e.g., social networks). Figure 1 shows the performance of the state-of-the-art shortest path distance solutions (cf. [5, 6, 9, 10, 18, 21, 23, 38, 40]) on two types of networks, CAL (road network, 1.8M vertices and 2.3M edges) and SKITTER (complex network, 1.7M vertices and 11M edges). Except TNR [10], none of the network traversal based solutions can compete with HL solutions, e.g., DHP [6], PHL [5], BHP [16], and SHP (this work), in terms of response time. Although there exist some experimental studies [3, 5, 6, 18] on HL techniques, we identify four issues in these studies.

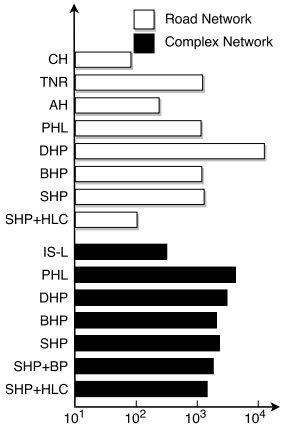
**[Experimental Settings]** We summarize the discrepancies among their experimental settings as follows. Thus, it is hard to infer a clear winner among the HL techniques [3, 6, 5, 18].
($a$) Experimental results were conducted on different platforms which might misreport the relative performance, e.g., competitors on Linux vs. own solution on Windows [16], competitors on Windows vs. own solution on Linux [5].
($b$) The size of the data structures were not reported, e.g., [6] mentioned they used 8-bit for distance attribute while [3,


*Proceedings of the VLDB Endowment,* Vol. 11, No. 4

DOI: https://doi.org/10.1145/3164135.3164141




(a) Response time vs indexing time




(b) Index size (MB)

| | Property | | Type | |
|---|---|---|---|---|
| | Directed | Weighted | Road | Complex |
| CH [23] | ○ | ○ | ○ | △ |
| AH [40] | ○ | ○ | ○ | × |
| TNR [10] | ○ | ○ | ○ | × |
| PHL [5] | × | ○ | ○ | △ |
| IS-L [21] | ○ | ○ | △ | ○ |
| DHP [6, 32] | ○ | ○ | △ | ○ |
| BHP [16] | ○ | ○ | ○ | ○ |
| SHP | ○ | ○ | ○ | ○ |
| +BP [6] | ○ | × | × | ○ |
| +HLC [18] | ○ | ○ | ○ | ○ |

(c) Applicable? ○: yes, △: yes but too costly, ×: no

**Figure 1: Relative performance of the shortest path distance on networks of ∼1.5 million vertices**

16] did not elaborate it in their work. For HL methods, the size of distance attribute significantly affects the index size and memory usage.
(*c*) Some performance factors were not reported, e.g., no response time for DHP in [6, 16], no indexing time for compression techniques in [16].

**[Completeness of Evaluation]**
(*a*) Solutions were compared only on a specific network type, e.g., only undirected complex networks in [6, 16] and only road networks in [5]. The performance of solutions on other network types remains unknown.
(*b*) The performance comparison between the state-of-the-art network traversal solutions and the latest HL techniques remains unclear. For instance, AH [40] can be viewed as the state-of-the-art solution on road networks but there is no experimental comparison with the latest HL techniques. In addition, most HL studies focused on distance queries but did not report the performance of path queries.

**[Hub Labeling Optimization]** Existing work [3, 6, 16, 18, 28] treat the hub labeling construction as a holistic algorithm, which may conceal the contribution of each individual technique. For instance, a fast label construction optimization is proposed in [6] but it has not been considered in more recent studies [16]. Besides, some nice properties from the path-based solutions [5, 29, 35, 39] could be extended to improve the HL techniques.

**[Performance Factors]** The performance factors include the response time, the index size, and the indexing time. Some techniques excel in one performance factor but perform badly in others. We are not aware of any unified metric for comparing the performance between the solutions.

## 1.2 Contributions

**[Unified Experimental Settings]** In order to fairly evaluate the methods, we conducted our experiments on 22 representative datasets which cover various graph categories in real world. By observing experimental results on different types of graphs with a great variety of graph properties, we obtain several insightful conclusions on how well a method can perform in a comprehensive set of scenarios. In addition, we have made necessary adaptations (e.g., using the same size for distance attribute) in implementations in order to enforce consistency.

**[Enriched Evaluation]** We systematically evaluate the HL techniques [3, 6, 16] with all known optimizations [6, 16, 18]. In addition, this is the first work to compare the latest HL techniques with other state-of-the-art competitors (e.g., AH [40]).

**[Significant Path Optimization]** A significant path based ordering heuristic (cf. Section 4.3) is proposed for improving the index construction of HL, which is inspired by path based solutions [5, 29, 35, 39]. Our new ordering heuristic greatly improves the indexing time while retains good query performance. More importantly, it achieves scalability for extremely large datasets which cannot be supported by the previous techniques [6, 16].

**[Overall Performance]** We propose a metric, called Overall Performance Score (OPS) (cf. Section 6.3.4), to express the overall performance of a method. This helps to decide the best all-round method in various application scenarios. In addition, we define the Preference Probability that compares the methods on different preference space. Lastly, we also give a summary that guides users to select the best method for their applications (cf. Section 7).

**[Source Codes]** The source codes of our implementations can be accessed at [1].

The remainder of the paper is organized as follows. Section 2 elaborates taxonomy of HL techniques. State-of-the-art construction paradigms and ordering schemes are introduced in Section 3 and Section 4 respectively, with detailed discussions on design philosophy. We briefly summarize optimization techniques in Section 5. Our extensive experimental analysis is demonstrated in Section 6. At last, with comparisons among HL techniques and against other state-of-the-art shortest path solutions, we summarize our suggestions and conclusions in Section 7.

# 2. PRELIMINARIES

## 2.1 Shortest Path and Shortest Path Distance

Let $G = (V, E)$ be a graph associated with a non-negative weight function $w : E \mapsto R_{\geq 0}$. Given a source $s$ and a destination $t$, the shortest path $SP_{s \to t}$ is the path from $s$ to $t$ with the smallest total distance (with respect to function $w$). We denote the shortest path distance, i.e., the total distance of $SP_{s \to t}$, by $dist(SP_{s \to t})$.

## 2.2 Hub Labeling

### 2.2.1 Background of Hub Labeling

DEFINITION 1 (HUB LABEL OF VERTEX $v$). *A hub label of a vertex $v$ is a key/value pair that consists of a hub indicator $h$ and the distance between $v$ and $h$.*

Definition 1 shows the formal definition of a label in the HL techniques. For undirected graphs, a hub $h$ can be viewed as a transit vertex between $v$ and other vertices. We denote the label set of a vertex $v$ as $L(v)$. For directed graphs, the hub labeling index stores two label sets of every vertex, where one set stores the distance information from $v$ to other vertices (forward labels) and another set stores the distance from other vertices to $v$ (backward labels). For ease of our discussion, we use undirected graphs by default. We will explain if there is any difference to apply HL techniques for directed graphs.

We summarize our frequently used notations.

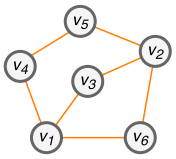
| symbol | description |
|---|---|
| $G = (V, E)$ | a graph with vertices $V$ and edges $E$ |
| $SP_{s\to t}$ | the shortest path from $s$ to $t$ |
| $dist(SP_{s\to t})$ | the distance of $SP_{s\to t}$ |
| $L(v)$ | the label set of $v$ |

**Table 1: Notations**

### 2.2.2 Label Cover Property

PROPERTY 1 (LABEL COVER). *For each pair of reachable vertices $s, t \in V$, there exists at least one shortest path vertex $h \in SP_{s\to t}$ in both label sets, $L(s)$ and $L(t)$.*

To answer the shortest path distance of any two vertices, the hub labels must fulfill a *cover property* that guarantees the shortest path distance of any two reachable vertices can be found in a sort-merge join between the label sets of two vertices. (Property 1)




(a) Graph $G$

| | $L(v)$ |
|---|---|
| $v_1$ | $\underline{v_1{:}0(-)}$ |
| $v_2$ | $\underline{v_1{:}2(v_3)}$, $v_2{:}0(-)$ |
| $v_3$ | $v_1{:}1(v_1)$, $v_2{:}1(v_2)$, $v_3{:}0(-)$ |
| $v_4$ | $\underline{v_1{:}1(v_1)}$, $v_2{:}2(v_5)$, $v_3{:}2(v_1)$, $v_4{:}0(-)$ |
| $v_5$ | $\underline{v_1{:}2(v_4)}$, $v_2{:}1(v_2)$, $v_4{:}1(v_4)$, $v_5{:}0(-)$ |
| $v_6$ | $\underline{v_1{:}1(v_1)}$, $v_2{:}1(v_2)$, $v_6{:}0(-)$ |

(b) A possible hub labeling of $G$

**Figure 2: A hub labeling index fulfilled the cover property (the brackets indicate the predecessors)**

Figure 2 shows a graph $G$ and a possible hub labeling index that fulfills the cover property. For instance, $v_2$ is kept in $v_5$'s label set $L(v_5)$ and $v_6$'s label set $L(v_6)$ since $v_2$ falls on the shortest path $SP_{v_5\to v_6}$.

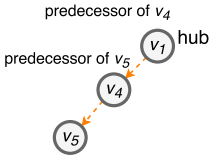
### 2.2.3 Hub Labeling Query Processing

**Distance queries.** Given the hub labeling index $L$ of a graph, the shortest path distance from $s$ to $t$ can be answered by

$$dist(SP_{s\to t}) = \min_{h\in L(s)\cap L(t)} \{dist(SP_{s\to h}) + dist(SP_{h\to t})\}$$

The above equation can be computed by a sort-merge join in linear time with respect to the sizes of the label sets. The time complexity is $O(|\bar{L}|)$ where $|\bar{L}|$ indicates the average label size of all label sets. According to [36], the distance query is particularly helpful in estimating the strength between two users in a social network.

**Path queries.** In order to trace back the shortest path from the hub labels, we keep the vertex's predecessor for each hub label, e.g., the brackets in Figure 2(b). For instance, given a hub label $v_1 \in L(v_5)$, we say $v_4$ is the predecessor of $v_5$ in a tree rooted at hub $v_1$. Based on the predecessor information, we can backtrack the path by a recursive process. As an example of $SP_{v_5\to v_1}$, we first join the label sets of two vertices and the joined hub is $v_1$ (underlined labels in Figure 2(b)). Based on the label in $L(v_1)$, we know $v_1$ is the end vertex in the path since it has no predecessor (denoted as "$-$" in Figure 2(b)). Based on the label in $L(v_5)$, we know the predecessor of $v_5$ (towards hub $v_1$) is $v_4$. Next we backtrack $L(v_4)$ and find $v_1$ as the predecessor of $v_4$ (in a graph rooted at $v_1$). Finally, we reconstruct the shortest path $v_5 \to v_4 \to v_1$ as shown in Figure 3. The complexity is $O(|\bar{L}| \cdot |\bar{SP}|)$ where $|\bar{SP}|$ indicates the average length of the shortest paths.




(a) Graph rooted at $v_1$

| labels | hub | prec. | shortest path |
|---|---|---|---|
| $L(v_1)$ | $v_1$ | $-$ | $\ldots \to v_1$ |
| $L(v_5)$ | $v_1$ | $v_4$ | $v_5 \to \ldots \to v_1$ |
| $L(v_4)$ | $v_1$ | $v_1$ | $v_5 \to v_4 \to v_1$ |

(b) Path reconstruction steps

**Figure 3: Path reconstruction for $SP_{v_5\to v_1}$**

### 2.2.4 Canonical Hub Labeling

From theoretical point of view, the optimal hub labeling should provide the best query performance (i.e., response time) as it has the minimum average label size. However, constructing the optimal hub labeling is a non-trivial NP-hard problem [15]. A polynomial-time algorithm having an approximation ratio $O(\log |V|)$ has been proposed by Cohen et al. [15]. This approximate algorithm takes $O(|V|^4)$ time and $O(|V|^2)$ space.

The prohibitive costs of the approximate algorithm call for other directions to address the scalability issues. Many subsequent work [2, 3] attempt to construct the hub labeling based on a heuristic idea called *network hierarchy*.

The network hierarchy has been imposed in many modern shortest path solutions, e.g., highway hierarchy (HH) [4], contraction hierarchy (CH) [2, 23], etc. Roughly speaking, a vertex $v$ (an edge $e$) is more important than another vertex $v'$ (edge $e'$) in a network if $v$ ($e$) exists in more pair-wise shortest paths than $v'$ ($e'$). As an example in road networks,

a highway segment is more important than a residential segment since a highway is the transit point between many pairs of sources and destinations.

Definition 2 (Canonical hub labeling). *Given a total order (i.e., hierarchy) $\mathcal{O}$ of all vertices $V$, a canonical hub labeling is the labeling that contains only the following labels: for every shortest path $SP_{s\to t}$, the highest ranked vertex $v \in SP_{s\to t}$ is kept in the label sets $L(s)$ and $L(t)$.*

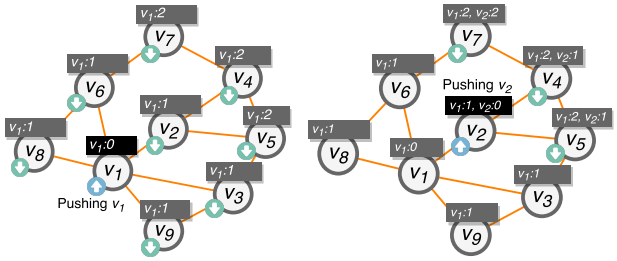
Abraham et al. [3] is the first work to build the hub labeling based on the network hierarchy. To impose the network hierarchy, a new definition, called canonical hub labeling (see Definition 2), is defined in [3] that restricts the appearance of the labels.

We can derive three properties from Definition 2. (1) The canonical hub labeling satisfies the cover property since it keeps at least one common vertex for each shortest path. (2) In addition, a less important vertex cannot be the hub label of a more important vertex since only the highest ranked vertex of a shortest path $SP_{s\to t}$ is kept in $L(s)$ and $L(t)$. (3) Lastly, the canonical hub labeling is minimal since removing any hub $h$ from a label set will violate the cover property. This is because $h$ is the highest ranked vertex of a shortest path $SP_{s\to t}$.

Even though the canonical hub labeling secures the cover property and minimality (i.e., no redundant label existed), the performance remains uncertain as the size of the canonical hub labeling is sensitive to the total order $\mathcal{O}$.

## 3. CONSTRUCTION PARADIGM

In this section, we first focus on the construction paradigm of the canonical hub labeling. Given a total order $\mathcal{O}$, the construction paradigm is to build a labeling that fulfills Definition 2. Without explicitly mentioning the graph type, we assume that the default graph is *undirected* and *unweighted*. We will add the discussion of the *directed* and *weighted* cases if there is any difference from the default case.




(a) Pushing hub $v_1$ to all vertices (b) Pushing hub $v_2$ to $v_4$, $v_5$, $v_7$

**Figure 4: Hub pushing algorithm**

To build a canonical hub labeling, one idea is to push the hub information from higher ranked vertices to lower ranked vertices such that (**O1**) all shortest paths are covered, (**O2**) the highest ranked vertex of any shortest paths is kept, and (**O3**) the labeling is minimal. The most common construction is called *hub pushing*, which iteratively pushes a vertex (as a hub label) to all its reachable vertices of lower rank. For instance, the highest ranked vertex $v_1$ is pushed to all vertices in Figure 4(a), where the label distances are set to their network distances. Obviously, this pushing mechanism fulfills objectives (**O1**) and (**O2**) but fails to secure the minimality (**O3**).

To address the minimality (**O3**), Akiba et al. [6] proposed a pruning method, called Pruned Landmark Labeling (PLL), where a label $v$ stops pushing further at a vertex $w$ if it has been *covered* by any prior label(s) in $L(w) \cap L(v)$. Specifically, suppose $h$ is a co-existed label in both $L(w)$ and $L(v)$, the pushing vertex $v$ is being covered by $h$ if $dist_{v\to w} \geq dist_{v\to h} + dist_{h\to w}$. In other words, hub $h$ can be used to answer all shortest paths from $v$ to $w$ and beyond so we stop pushing $v$ at $w$.

As shown in Figure 4(b), $v_2$ stops pushing at vertices $v_3$ and $v_6$ since it has been covered by hub $v_1$, i.e., $dist_{v_2\to v_3} \geq dist_{v_2\to v_1} + dist_{v_1\to v_3}$ and $dist_{v_2\to v_6} \geq dist_{v_2\to v_1} + dist_{v_1\to v_6}$. Suppose we want to calculate the shortest path from $v_2$ to $v_8$, the co-existed hub $v_1$ (in both $L(v_2)$ and $L(v_8)$) can be used to answer the query.

**Algorithm 1** HubPushing($G$: graph; $\mathcal{O}$: vertex order)

```
   Q: Queue
1: for v ∈ V subject to the order of O do ▷ Pushing v as a hub
2:     Q = {(v, 0)}                                  ▷ BFS from v
3:     while Q is not empty do
4:         (w, dist) = Q.pop()
5:         d = min_{h∈L(v)∩L(w)} {dist_{v→h} + dist_{h→w}}
6:         if dist < d then                ▷ Pruned by co-existed label
7:             Add label (v, dist) into L(w)
8:             for unvisited neighbor vertex u of w do
9:                 Q.push(u, dist + 1)
```

For clarity, we show the pseudo codes of the hub pushing algorithm in Algorithm 1. For each vertex $v$ (subject to the vertex order $\mathcal{O}$), we run a network traversal (e.g., BFS[1]) from $v$. Line 5 returns the shortest path distance using the hub labels (cf. Section 2.2.3). We stop further traversal at a vertex $w$ if there is a common hub $h$ in $L(v)$ and $L(w)$ such that $dist_{v\to w} \geq dist_{v\to h} + dist_{h\to w}$ (cf. Line 6). Otherwise, the network traversal continues and $v$ is added into the label set of $w$ (cf. Lines 7-9).

**Directed graphs.** In directed graphs, two network traversals are conducted from each vertex $v$. Forward (backward) network traversal from $v$ pushes it into backward (forward) label sets of reachable vertices.

**Alternative: Hub pulling algorithm.** Abraham et al. [2] proposed a hub label construction method that *pulls* the label sets from all reachable vertices of higher ranks. As an example in Figure 4, when constructing $L(v_2)$, we pull the labels from $L(v_1)$ as $v_1$ is the only reachable vertex of higher rank. However, this construction algorithm is no longer considered in recent studies [6, 16, 28] due to the cost of finding all reachable vertices, where an experimental comparison can be found in [6].

## 4. ORDERING SCHEME

Given a graph $G$, the construction algorithm (e.g., Hub Pushing Algorithm) builds a unique canonical hub labeling of the same **index size** subject to a vertex order $\mathcal{O}$. Given that the response time of a shortest path distance query is linear to the label size, the **query performance** is indeed

[1]For a weighted graph, BFS should be replaced by a Dijkstra like algorithm.

sensitive to $\mathcal{O}$. The **construction time** is also correlated to the quality of $\mathcal{O}$ since the pruning processing takes linear time of the label size. It is indisputable that the ordering scheme is decisive in the canonical HL techniques.

In this section, we will introduce two widely used ordering schemes, DHP (Degree based Hub Pushing) [6, 28, 32] and BHP (Betweenness based Hub Pushing) [2, 3, 8, 15, 16], which rank vertices based on local and global information, respectively. In addition, we also study a simple yet effective ordering scheme, SHP (Significant path based Hub Pushing), that shows a better trade-off between the indexing time and the ordering quality in practice.

## 4.1 Degree based Hub Pushing (DHP)

Vertex degree is considered as a natural indicator for the importance of a vertex in many shortest path solutions [21, 32, 37]. This simple yet effective ordering scheme is adopted in canonical HL techniques [6, 28]. Jiang et al. [28] was the first attempt to study the robustness of the degree ordering in the canonical hub labeling. They observed that a vertex of high degree is probably a good hub for many other vertices. However, our experiments (as well as [16]) demonstrate that the degree ordering scheme may yield larger label size than other schemes due to the lack of use of global information.

**Directed graphs.** As suggested in [28, 39], we may simply use the multiplication of the in-degree and the out-degree of a vertex to represent its degree score. Their experiments show that this idea exhibits good performance in practice.

## 4.2 Betweenness based Hub Pushing with Sampling Technique (BHP)

In graph theory, the betweenness of a vertex $v$ is the number of shortest paths passing through $v$. Accordingly, a vertex of high betweenness can be viewed as an important vertex since it is a hub that can answer many shortest paths. Recent canonical HL studies [2, 3, 8, 15] have already employed the betweenness [14, 22] in ordering. For instance, Abraham et al. [2, 3] compute the betweenness using shortest path trees. Suppose every shortest path is unique, the betweenness of a vertex $v$ is the sum of the descendant sizes of $v$ in all shortest path trees.

The betweenness ordering scheme works with the construction algorithm as follows. At each hub pushing iteration (cf. Line 2-9 of Algorithm 1), the vertex of the highest betweenness (e.g., $v$) is selected as the hub. Then the descendants of $v$ are removed from all shortest path trees since hub $v$ is sufficient to answer the shortest path from any descendant to the root. The betweenness of all unselected vertices are then recomputed. However, this solution requires quadratic space to store the shortest path trees.

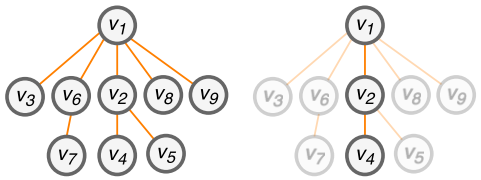
To address the quadratic cost, random sampling [14, 22] is then introduced to boost the betweenness computation. Delling et al. [16] proposed to use only $s_k$ shortest path trees as a sampled set $T$. The betweenness of each vertex is then estimated based on $T$. However, the estimation based on $T$ becomes less accurate at later iterations since many descendants of the trees have been removed. The estimation accuracy can be recovered by building extra shortest path trees (being controlled by a parameter $s_\beta$) subject to a space budget $s_c$. These newly sampled trees are then added into $T$ for enhancing the estimation quality. It stops sampling more trees when the size of $T$ (i.e., the number of vertices) exceeds $s_c \cdot |V|$. According to [16], the default settings ($s_k = 16, s_\beta = 1, s_c = 160$) show a robust performance.

## 4.3 Significant path based Hub Pushing (SHP)

Typically BHP is more costly to construct than DHP since it requires to maintain $|T|$ sampling trees. When the vertex degrees are insufficient to identify the importance of vertices, DHP may be more costly to construct. This motivates us to ask a question: *Can we propose a more systematic solution that preserves good ordering (e.g., generating small $|\bar{L}|$) but has relatively low overhead (e.g., removing cost factor $|T|$)?*

Our solution is inspired by path based (i.e., multi-hops) HL techniques, e.g., PHL [5] and [29, 39], where the hub label indicator becomes a path (instead of a vertex) in a label. A common observation in this work is that a shortest path $P_{sig}$ is significant if $P_{sig}$ (and the sub-paths of $P_{sig}$) is passed through by many other shortest paths. Apparently, a vertex $v_{sig} \in P_{sig}$ should also be significant as the betweenness of $v_{sig}$ is relatively high.

To find a significant path of a graph, PHL [5] starts from an important vertex $v$ (e.g., the highest degree vertex) and builds the shortest path tree rooted at $v$. We observe that the hub pushing process (lines 2-9) in Algorithm 1 is also returning a shortest path tree rooted at $v$ (i.e., an important vertex according to $\mathcal{O}$).



(a) SPTree of $v_1$ (b) $P_{sig}$

**Figure 5: Identifying a significant path**

After the shortest path tree is built, PHL recursively (from root to leaf) picks a child vertex of the largest descendants on $P_{sig}$. Figure 5 shows a concrete example, where $v_2$ is picked as it has more descendants than other vertices on $P_{sig}$. Our next mission is to pick the significant vertex from $P_{sig}$, which will be used as the next hub to be pushed in Algorithm 1.

Note that the significant vertex $v_{sig} \in P_{sig}$ cannot be picked according to the descendant size since some vertices in the shortest path tree are already covered by root $v$ (i.e., adding $v_{sig}$ into the label set of these vertices does not help to answer any shortest path queries). To effectively pick a significant vertex $v_{sig} \in P_{sig}$, we suggest to rank the vertices using two common heuristics, *vertex degree* and *size difference of descendants* [2]. For simplicity, we suggest to pick $v_{sig}$ based on the multiplication of these two values.

**Discussion.** In short, the significant path should provide better ordering than the vertex degree since the order is decided not only by the neighborhood information but also by the descendants. It offers reasonable overhead as compared to the betweenness based solution. We will extensively compare these ordering techniques in the experimental section.

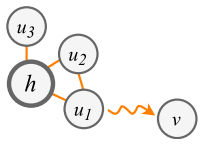
[2]For instance, the descendant size difference of a highway vertex (in road networks) is high since there are many residential vertices connected to it.

# 5. OPTIMIZATION TECHNIQUES

In this section, we discuss an optimization on how to reduce the indexing time in Section 5.1, and then two optimizations on how to reduce the index size in Section 5.2.

## 5.1 Fast Label Construction

The first optimization, called Bit-Parallel (BP) [6], is only applicable to unweighted graphs. To reduce the label construction time, BP executes hub pushing for multiple neighbor hubs simultaneously and also stores some hub labels in compact bit vectors.




(a) BP pushing

| | Pushing $h$ to $v$ |
|---|---|
| $L(v)$ | $h:2$ |
| $B_h^0(v)$ | "010": meaning ~~$u_1$~~ $u_2$ ~~$u_3$~~ |
| $B_h^{-1}(v)$ | "100": meaning $u_1$ ~~$u_2$~~ ~~$u_3$~~ |

(b) BP hub labeling

**Figure 6: BP hub labeling index of $G$, using 3-bit vectors**

Consider the example in Figure 6 and assume that the vertex order is: $h, u_3, u_2, u_1, \cdots$. The original hub pushing algorithm processes each hub one-by-one, then obtains the label set of $v$ as $L(v) = \{h:2, u_2:2, u_1:1\}$. With the BP technique, the hub $h$ and its set of neighbor hubs (e.g., $S_h = \{u_1, u_2, u_3\}$) will be processed simultaneously. For an unweighted graph, the distances $dist(SP_{u_i \to v})$ and $dist(SP_{h \to v})$ must differ by either -1, 0 or 1. With this observation, BP encodes the vertices of $S_h$ as three compact bit vectors, instead of storing them in $L(v)$. In the bit vector $B_h^j(v)$, the $i$-th bit is set if $dist(SP_{u_i \to v}) - dist(SP_{h \to v}) = j$. For example, the bit vector $B_h^{-1}(v)$ = "100" means that $dist(SP_{u_1 \to v}) - dist(SP_{h \to v}) = -1$. In practical implementation, BP limits the size of $S_h$ to 64 so that each bit vector can be encoded into a 64-bit unsigned integer.

During query processing between $s$ and $t$, BP can retrieve the hub information by using both the label sets $L(s)$, $L(t)$, and their bit vectors. Thus, the query time is asymptotically identical to the original hub index [6]. In some cases, BP also improves query performance due to smaller label set for each vertex.

**Applicability.** BP can only support unweighted graphs. When it is applied to directed graphs, the parallel processing performance will be degraded since only the neighbors coexist in both in/out-neighbor lists can be taken into $S_h$.

## 5.2 Label Compression

Hub Label Compression (HLC) [18, 16] compresses hub labeling by indexing and referencing common groups of hub labels. We illustrate common hub vertices with an example. First, in Figure 4(a), $v_1$ is pushed to $v_2$, and then later pushed to $v_4$ and $v_5$. In the next iteration (see Figure 4(b)), $v_2$ is also pushed to $v_4$ and $v_5$. Hence $L(v_4)$ and $L(v_5)$ contain common hubs $v_1$ and $v_2$.

HLC represents common groups of hubs as subtree structures [18]. We show the framework to compress a hub label set $L(v)$ in Figure 7. First, HLC transforms each hub label set $L(v)$ into a tree $T_v$ rooted at $v$ [2], where each tree node represents a hub in $L(v)$. A tree edge $(h_1, h_2)$ is inserted if the shortest path between two hubs $h_1, h_2$ does not pass any other vertex in $L(v)$. Then, HLC compresses the tree in a bottom-up fashion as follows. It replaces a two-layer subtree with a leaf node as a token (shown in black color). A hash module is used to generate the token ID and to guarantee that the tokens are unique. The tokenization continues until the entire tree is compressed to a single token (tree node). When HLC processes the hub label sets for other vertices, existing tokens (and subtrees) can be reused to reduce the space.

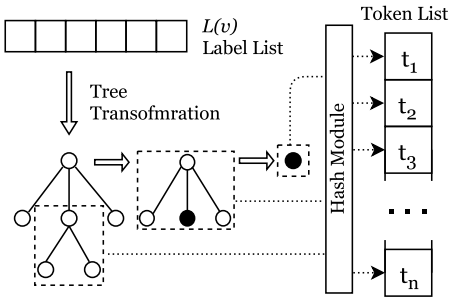



**Figure 7: Tokenization of a label set**

While HLC provides effective compression for road networks, it is less effective for complex networks since subtrees in complex networks tend to be wide and shallow. Delling et al. [16] extend this method to HLC with masks (HLCM), in which different tokens with same root are merged into a supertoken and a single token is represented by a bit-reference to its supertoken.

**Query.** At the query time, a traversal algorithm can be applied on compressed lists to rebuild the original label sets. After that, the hash-join operation is invoked to compute the intersection of hubs and thus the distance [18].

**Applicability.** Both HLC and HLCM can support any type of graphs. However, they cannot be used for path query since predecessor information can not be compressed.

# 6. EXPERIMENTS

This section presents our experimental comparisons and analysis for all canonical HL techniques as well as the state-of-the-art competitors, including CH [23], TNR [10], AH [40], IS-L [21], PHL [5].

## 6.1 Experimental Settings

We implemented all discussed HL techniques, including three hub pushing algorithms (DHP, BHP, and SHP) in Section 4, and two optimization techniques (fast label construction in Section 5.1, and label compression in Section 5.2). We have released our source codes at [1]. Regarding the competitors (CH, TNR, AH, IS-L, PHL), we obtained their source codes from the authors [5, 21, 40] and a previous experimental study [38], then made necessary adaptation to enforce consistency with our experimental settings.

All algorithms were implemented with C++ and compiled by g++ 4.8 with -O3 flag. All experiments were conducted on a Linux Sever in 64-bit Ubuntu 14.04.2 LTS with Intel Xeon E5-2650v3 and 256GB main memory. We omitted the result of a method if it ran out of memory or did not terminate within 10 hours.

Regarding the parameter settings in the competitors, we follow exactly the same suggestions from their authors. For

instance, we set $s_k = 16, s_\beta = 1, s_c = 160$ for BHP [16], the number of parallel iterations to 50 for Bit-Parallel [6], and level parameters to 215 and 6 for graphs smaller and larger than 1 million vertices for IS-L, respectively [21]. For road network solutions (e.g., CH [23], TNR [10], AH [40]), we follow the settings of prior studies [38, 40].

| Graph | | Name | Category | $\|V\|$ | $\|E\|$ | degree | diameter |
|---|---|---|---|---|---|---|---|
| Complex | Directed | GNUTELLA | Social | 62K | 147K | 4.7 | 11 |
| | | SLASHDOT | Social | 82K | 1M | 11.5 | 11 |
| | | NOTREDAME | Hyperlink | 325K | 2M | 9 | 46 |
| | | DBLP | Citation | 1.3M | 37M | 7.9 | 10 |
| | | WIKI-POLISH | Hyperlink | 1.5M | 11M | 75.2 | 10 |
| | | TRECWT10G | Hyperlink | 1.6M | 16M | 10.1 | 112 |
| | | WIKITALK | Comm. | 2.3M | 5M | 2.1 | 9 |
| | | FLICKR | Social | 2.3M | 33M | 28.8 | 23 |
| | | INDOCHINA | Hyperlink | 7.4M | 383M | 52 | 207 |
| | | UK2002 | Hyperlink | 19M | 584M | 32 | 183 |
| | Undirected | CATDOG | Social | 62K | 15M | 50.3 | 15 |
| | | COM-DBLP | Citation | 317K | 1M | 6.6 | 23 |
| | | SKITTER | Computer | 1.7M | 11M | 13.1 | 31 |
| | | YOUTUBE | Social | 3.2M | 9M | 5.8 | 31 |
| | | HOLLYWOOD | Social | 1.1M | 112M | 99 | 9 |
| Road | Undirected | BAY | Area | 321K | 400K | 1.25 | 1311 |
| | | FLA | State | 1.1M | 1.3M | 1.27 | 2826 |
| | | CAL | State | 1.8M | 2.3M | 1.23 | 4482 |
| | | E | Region | 3.5M | 4.4M | 1.22 | 6824 |
| | | W | Region | 6.2M | 7.6 M | 1.22 | 8085 |
| | | CTR | Region | 14M | 17M | 1.22 | 8597 |
| | | USA | Continent | 24M | 29M | 1.22 | 13941 |

**Table 2: List of datasets**

**Datasets.** Our testing datasets cover different categories of real-world graphs in order to conduct a complete and fair evaluation. These datasets can be broadly classified into two major types, namely complex networks and road networks, as shown in Table 2. All datasets are accessible at [19, 30, 31, 13].

**Queries.** We evaluate the shortest path distance queries on both complex and road networks. However, we only evaluate the shortest path queries on road networks because the shortest paths on complex networks are short in practice. The performance of the path queries is similar to that of the distance queries on complex networks.

## 6.2 Performance Factors

**Indexing time.** The indexing time denotes the CPU clock time for the index construction. The I/O time of building the index file(s) is excluded as in previous studies [38, 2, 3, 6, 16].

**Response time.** We report the average response time of 1 million random queries. Prior to running experiments, 0.5 million queries are used for warm-up.

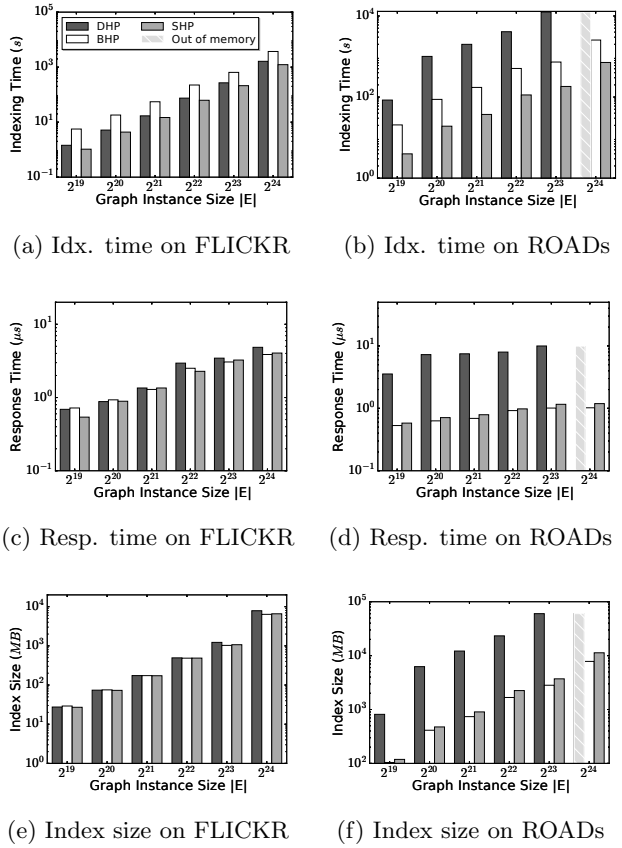
**Index size.** We use a 32-bit integer to represent a vertex ID or a distance value in the index. We report the index size for each evaluated method. For HL techniques, each label is a 2-tuple $\langle hub, dist\rangle$ for those solutions only applicable to distance queries (i.e., SHP+BP and SHP+HLC) and a 3-tuple $\langle hub, dist, predecessor\rangle$ otherwise.

## 6.3 Evaluation of HL Techniques

In this subsection, we extensively evaluate all discussed HL techniques, including ordering schemes, construction paradigms and optimizations.

### 6.3.1 Ordering Schemes

Recall that we have examined three ordering schemes in Section 4, namely, degree based (DHP), betweenness based (BHP), and significant path based (SHP).




(a) Idx. time on FLICKR (b) Idx. time on ROADs

(c) Resp. time on FLICKR (d) Resp. time on ROADs

(e) Index size on FLICKR (f) Index size on ROADs

**Figure 8: Scalability of ordering schemes**

**Scalability of ordering schemes.** First, we study the scalability of ordering schemes by varying the graph instance size from $2^{19}$ edges to $2^{24}$ edges (i.e., from 0.52M edges to 16.7M edges). For complex networks, we extract 6 historical snapshots of FLICKR subject to their sizes, and denote this collection as FLICKR. For road networks, we pick BAY, FLA, CAL, E, W and CTR, and denote this collection as ROADs.

Figure 8 shows the performance of three ordering schemes. The indexing and response time of all methods increase proportionally with the network size. SHP offers the lowest indexing time since the significant path is relatively easy to compute. In addition, the response time of SHP is similar to BHP, e.g., BHP is only 4.7% faster than SHP on the largest instance. This indicates that SHP and BHP generate hub labels of similar size on different network instances.

Although DHP is the default ordering scheme in many HL studies [6, 28, 32], its performance is very sensitive to the network types. As compared to SHP, the indexing time of DHP is just 25% (on average) slower on FLICKR but 46 times (on average) slower on ROADs. This is because the degree of vertices in road networks is very small (e.g., 1.23 on average for all ROADs), that is ineffective to identify

important vertices. Besides, the average response time of DHP is 8 times slower than SHP on ROADs.

**Robustness of ordering schemes.** Next we verify the robustness of 3 ordering schemes by demonstrating their performance on 22 networks in Figure 9. SHP is consistently the fastest indexing method on all networks due to its light overhead and good ordering quality. Regarding the response time, SHP is competitive; it is only 10% slower than BHP on average. The performance of DHP is not robust across different networks. For example, DHP performs well on many complex networks but not on TRECWT10G. Thus, we do not recommend DHP due to its instability. For the largest network UK2002, both DHP and BHP run out of memory in the index construction process, whereas SHP can complete successfully.

**Effects of graph properties.** By referring to Table 2 and Figure 9, we study how graph properties (e.g., diameter, average degree) affect performance of different ordering schemes. For road networks, DHP is inferior to other two methods in label quality since the degree based ordering becomes ineffective in graphs of high diameters. For complex networks, we divide them into low-diameter instances (with diameter $\leq 20$) and high-diameter ones (with diameter $> 20$). The response time of DHP is one time slower than BHP on average in high-diameter instances while the superiority of BHP to DHP degrades to only 24% in low-diameter ones. In contrast, our proposed SHP is consistently around 10% slower than BHP in response time on average (including road networks, in which DHP is 10 times slower than BHP in response time). Meanwhile we do not observe any significant impact of average degree on response time of all three methods.

**Tuning of BHP.** SHP is parameterless whereas BHP requires a parameter $s_\beta$ in the sampling process. The higher the $s_\beta$ value, BHP yields a smaller label size but incurs a higher indexing time [16]. To investigate the effect of $s_\beta$, we tune $s_\beta$ from 0.1 to 1 (the default setting) with a step of 0.1. We conduct experiments on a complex network CATDOG and a road network FLA. Figure 10 plots the average label size and the indexing time of the methods. We label BHP (for different values of $s_\beta$) and SHP by circles and a triangle, respectively.

SHP consistently resides at the skyline trade-off points in both graphs, which indicates its efficiency in identifying good ordering. Furthermore, the sensitivity of $s_\beta$ is diverse in different graphs. Changes of $s_\beta$ dramatically affect label size of FLA due to the importance of highway dimension on road networks [4]. This incurs difficulties in tuning a universal $s_\beta$ for different graphs. Hence we recommend SHP as a trade-off solution rather than BHP.

### 6.3.2 Construction Paradigm

For completeness, we also demonstrate the indexing time of different construction paradigms in Figure 11. We assume all three orderings have been precomputed hence the indexing time of hub pushing only contains the time for label pushing processing, while hub pulling consists of overlay graph construction and label pulling processing. We parallelize overlay graph construction via OpenMP on 10 cores as suggested by [3].

The superiority of hub pushing is consistent for both types of networks on different network instances. For example, with significant path based ordering, hub pushing is on average 2 times and 1 time faster than hub pulling on FLICKR and ROADs respectively.

### 6.3.3 Optimizations

**Fast label construction.** Next we investigate the effectiveness of BP applied to ordering schemes on unweighted FLICKR. For better presentation, we only demonstrate indexing and response time for SHP and SHP+BP in Figure 12.

The result indicates BP can achieve considerable improvement for both indexing time and query performance. On average, it enhances indexing and response time of SHP by 2 times and 3 times, respectively. In contrast, BP only improves those of BHP by 1.4 times and 2.5 times (we do not show these results in the figures). The extra improvements for SHP comes from the fact that important vertices in complex networks tend to be highly reachable. Hence even an unimportant vertex is picked as a BP hub, its neighbor hubs may contain some important vertices. In this way, BP can compensate the defectiveness of worse vertex ordering.

Overall, BP is an effective optimization for speeding up indexing and response time for unweighed complex networks.

**Label compression.** We verify the effectiveness of HLC and HLCM when they are applied to ordering schemes. For completeness, we also demonstrate a simpler compression technique in [16] called delta compression (DELTA), which is based on gap representation of adjacent lists [13]. The index size and response time are demonstrated in Figure 13 for both FLICKR and ROADs. For better presentation, we only show the results for SHP, SHP+HLC, SHP+HLCM and SHP+DELTA. The results demonstrates that, DELTA can not compress the index as much as HLC and HLCM for the reasons that it can only reduce the size of hub IDs. Although DELTA does not deteriorate much on the query performance, we focus our discussions on HLC and HLCM, which can compress the index much effectively.

For ROADs, we observe that both HLC and HLCM can reduce label size significantly compared to the original labels and they tend to compress more on SHP due to worse vertex ordering generally incurs more duplicated subtrees. The result also indicates that HLCM is a worse trade-off for ROADs. Compared to HLC, HLCM can only slightly reduce 28% more index size on SHP. In contrast, it drastically degrades average query performance of HLC by 3 times.

For FLICKR, HLCM is more effective in compression compared to HLC. And the degradation of query performance of both compression techniques is highly sensitive to their compression ratio.

HLC is not as effective as HLCM on FLICKR due to larger average degree in complex networks generally. To verify this conclusion, we provide experimental results of SHP and SHP+HLC on YOUTUBE and W respectively. Both networks have similar number of edges and index size generated by SHP. However, the compression ratio of HLC on W is 2 times better than that on YOUTUBE. HLC can achieve more compression when the unique tokens can be referred by a large number of duplicated subtrees in label sets. However, larger average degree of complex network results in flat subtrees which makes tokens can not be refered by others since a trival difference will generate a new unique token.

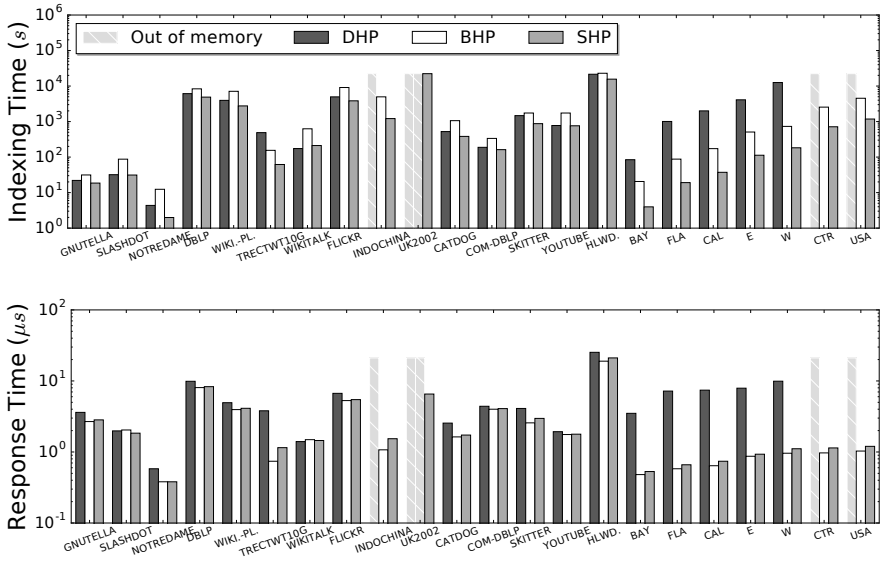
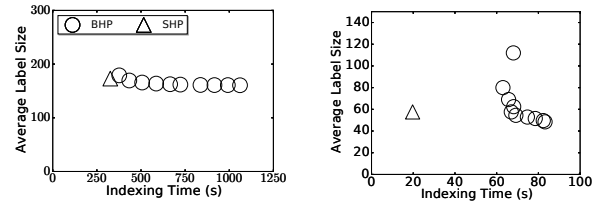
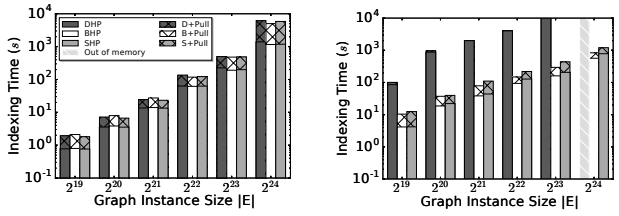



**Figure 9: Robustness of ordering schemes**



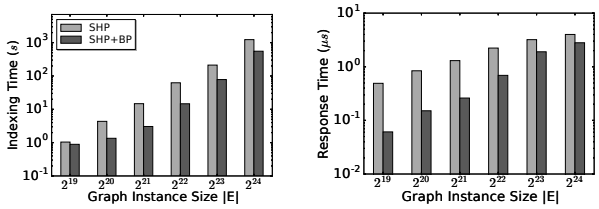

(a) Complex net CATDOG (b) Road net FLA

**Figure 10: Varying $s_\beta$ in BHP vs. SHP**




(a) Idx. time on FLICKR (b) Idx. time on ROADs

**Figure 11: Scalability of construction paradigms**




(a) Indexing time (b) Response time

**Figure 12: Effectiveness of BP on FLICKR**

We also verify that average degree has an impact on compression ratio of both HLC and HLCM in complex networks. We divide all 13 complex networks can be finished by all three methods into low-degree instances (deg: $\leq 20$) and high-degree ones (deg: $> 20$). And from the results, we observe that both of HLC and HLCM on average can achieve 2 times better compression ratio on low-degree instances than that on high-degree ones.

In summary, both HLC and HLCM can achieve compression ratio within one order of magnitude with sacrifice on query performance. Since HLCM is less robust in road networks, we recommend HLC as a better compression technique in consideration of both index size and response time.

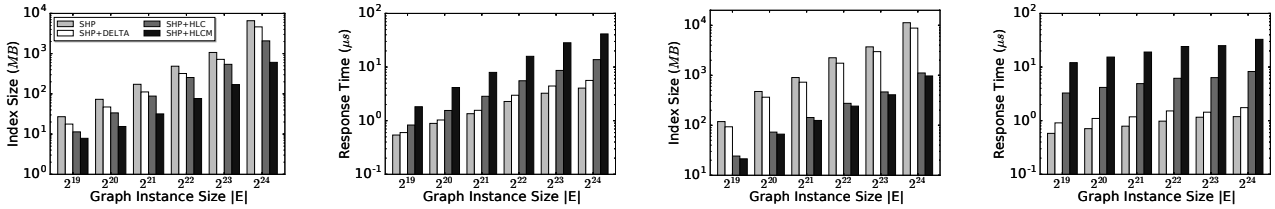
### 6.3.4 Overall Performance Score and Preference Probability

It is not easy to conclude the winner of all evaluated methods due to multiple performance factors. This motivates us to aggregate all performance factors into a unified score for fair comparison. Let $M$ be the set of all evaluated methods, $F$ be the target set of performance factors, and $D$ be the evaluated datasets, the relative performance of method $m \in M$ on dataset $d \in D$ in factor $f \in F$ among all evaluated methods can be represented as

$$score_{rel}(m, d, f) = \log_{10} \frac{score(m, d, f)}{\min_{m \in M} score(m, d, f)} \quad (1)$$

where $score(m, d, f)$ indicates the performance of method $m$ on dataset $d$ in factor $f$. Logarithm is used to normalize the relative performance. The method with smaller $score_{rel}$ score is preferable. The **Overall Performance Score (OPS)** of $m$ is defined as:

$$OPS(m, W, D, F) = \frac{\sum_{d \in D} \sum_{f \in F} w_f \times score_{rel}(m, d, f)}{|F||D|} \quad (2)$$

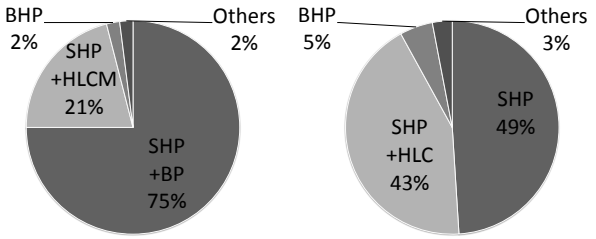



(a) Index size on FLICKR (b) Resp. time on FLICKR (c) Index size on ROADs (d) Resp. time on ROADs

**Figure 13: Effectiveness of HLC and HLCM**

where $W = \{w_f | f \in F\}$ denotes the the preference vector on factors (such that $\sum_{f \in F} w_f = 1$). For instance, assuming equal importance on three factors (i.e., $W = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$), the OPS of SHP for distance query on complex network NOTREDAME is 0.18, meaning that SHP is about on average 51.3% worse than the "ideal" all-round method in each performance factor.

In real applications, these three performance factors may not be equally important. For example, in the static scenario we may have heavier preference on response time, while in the dynamic scenario [17] we may rebuild the index frequently and have heavier preference on indexing time. We define the **Preference Probability** of a method $m$ by the probability of randomly sampling a preference vector $W$ (subject to $\sum_{f \in F} w_f = 1$) such that $m$ achieves the best OPS among all methods.




(a) Complex networks (b) Road newtorks

**Figure 14: Preference probability of HL techniques**

We show the preference probability of the evaluated HL techniques in Figure 14. We have the following conclusions: (1) SHP is consistently superior to BHP and DHP (with the same optimization) due to its high indexing efficiency and robust performance in query performance; (2) the compression techniques (e.g., HLC and HLCM) should be adopted if the index size is a concern in your applications; (3) BP can improve overall performance of each ordering scheme in unweighted graphs. In the next subsection, we only show experimental results for SHP, and its variants with HLC and BP optimization against other competitors.

## 6.4 Comparisons with Other Techniques

In this subsection, we compare HL techniques to other competitors for distance query application on both complex and road networks and for path query application on road networks. For complex networks, the competitors consist of PHL [5] and IS-L [21] while for road networks, the competitors include PHL, CH [23], TNR [10] and AH [40].

CH [23] is excluded for complex networks since the index cannot be built within a reasonable time (e.g., CH takes more than 10 thousands seconds in indexing a small complex network of ∼100k vertices [3].). TNR [10] and AH [40] are inapplicable to complex networks since both require spatial coordinates. IS-L [21] construction relies on a linear-time independent set heuristic which is only effective in complex networks. In addition, we exclude some HL optimizations for road networks (e.g., SHP+BP) and path queries (e.g., SHP+BP and SHP+HLC) due to their inapplicability.

All results are demonstrated in Table 3 (indexing time and index size), Table 4 (response Time). The empty slots indicate a specific method is not applicable or can not be finished within 10-hour running time or 256GB memory limits.

### 6.4.1 Complex Networks

We start our investigation by comparing HL techniques to PHL and IS-L.

**Compared to PHL.** To our surprise, even though PHL is specifically designed for road networks with highway structure, its query performance is competitive to other HL techniques. PHL can be considered as a multi-hops hub labeling [5], which utilizes intermediate paths as index to efficiently answer pairwise distance queries. Query processing of both PHL and HL techniques requires merge-join processing over sorted label. However, PHL requires two-layer join operation: firstly join by path hub and secondly join by entry vertices to retrieve relative distances on the path. When paths in complex networks are generally short, PHL cannot benefit from fast skipping in the first-layer join due to large number of path hubs. Hence 2-hop SHP outperforms PHL. Compared to SHP, PHL on average spends 8 times more indexing time with on average 50% larger index size. Also, another limitation of PHL is that it can only be applied to undirected graphs.

**Compared to IS-L.** IS-L can be considered as a combination method of graph traversal and hub labeling for complex networks[21]. It builds partial hub labeling to assist graph traversal. The level parameter balances indexing cost (indexing time and index size) and query performance. Hence with proper parameter, IS-L can be fast in indexing time with small index size (e.g., in TRECTWT10G). However, the graph traversal on complex network is expensive due to high density. Hence SHP, SHP+BP and SHP+HLC consistently outperform IS-L in terms of response time.

In summary, HL techniques exhibit efficient indexing, competitive index size and superior query performance compared to other techniques on complex networks.

### 6.4.2 Road Networks

We continue our investigation by comparing HL techniques to CH, AH, TNR and PHL on road networks.

**Indexing time.** We firstly study the relative performance on indexing time. Overall, CH, SHP and PHL are the most efficient methods with least indexing time while SHP and PHL are superior on small road networks and CH scales better on larger instances. Efficiency of SHP and PHL comes from their identification of important paths in road networks with few overhead. The efficiency of CH comes from its linear-time shortcutting processing in degree-bounded road networks with highway dimension [23, 4].

**Distance query.** The most efficient methods are PHL and SHP. The lengths of paths in road networks are generally long so that first-layer join in PHL contains only a few path hubs. Thus PHL can benefit from fast skipping in the first-layer join when compared to 2-hop hub labeling, e.g., SHP. We also observe that TNR is at high query efficiency since it imposes grids on road networks and utilizes pairwise index to directly retrieve distances between grids [10, 38], avoiding long-distance graph traversal. SHP+HLC is slightly slower than TNR, followed by AH and CH who are the least efficient methods in distance query since they involve extensive graph traversals [38, 40].

**Path query.** For the performance of path query, we observe quite different relative performance since all methods are required to partly traverse on the graph for path retrieval. Except TNR who incurs much slower path query response time, relative performance of other techniques are close to each other. Both SHP and PHL achieve faster path query response time in smaller road networks. However, CH and AH have better scalability in larger instances. In contrast to distance query, 2-hop SHP outperforms PHL in path query. Both methods requires $O(|\bar{L}| \cdot |\bar{SP}|)$ time for path retrieval and they can both benefit from early termination in each sorted label sets during their predecessor backtracking processing. However, SHP have better locality than PHL since the latter requires two-layer scans on two sets (path hubs and entry vertices) in predecessor checking. Path query of both CH and AH consists of graph traversals and shortcut unpacking [38, 40]. The shortcut unpacking processing only requires $O(|\bar{SP}|)$ constant time unpacking operations. And in generally the query of AH involves faster graph traversal and less shortcut unpacking operations than CH [40], providing AH the best scalability on path query among all techniques.

**Index size.** For index size, original HL techniques, e.g., SHP and PHL are around an order of magnitude larger than CH, TNR and AH who elaborate their designs for road networks with highway dimension. Although HLC cannot support path query application, it can greatly reduce the index size of HL techniques, making its storage very competitive even compared to the most lightweight CH, but still with an order of magnitude faster distance query efficiency than it.

Overall, for indexing time, distance and path query performance, HL techniques, e.g., SHP can be considered as the all-round skyline solution for road networks in all scale. With HLC compression optimization, HL techniques can achieve very lightweight index size but still offer efficient query performance.

### 6.4.3 Summary

HL techniques (e.g., SHP) can consistently achieve almost the best query performance for both distance and path query in both network types. With HLC optimization, the index size of HL techniques can be greatly reduced but they can still retain much higher query performance. Another important property of HL techniques is that they are widely applicable to all types of graphs with consistent performance.

## 7. CONCLUSION

We give our overall recommendation in Table 5. We recommend the top-3 methods according to their preference probability on three application scenarios (cf. Section 6.3.4), including distance queries on complex networks, distance queries on road networks, and path queries on road networks. We consider SHP, DHP, BHP, their optimized versions, and other competitors listed in Table 3 and 4 in the overall recommendation.

**Complex networks.** With the fast indexing time and good response time, SHP is the most recommended method for distance queries on complex networks. In addition, the runner-up method is SHP+HLCM, that is recommended when the index space is a concern. BHP is ranked at the third position which indicates the importance of finding a good vertex ordering.

**Road networks.** For distance queries, HL techniques are highly recommended since they support the distance query without any network traversal. Also, SHP+HLC is recommended when the size is a concern in the system. For path queries, the query response time of HL increases since it unavoidably traverses the labels in order to return a path. Thereby, CH becomes the most recommended method due to its fast construction cost and almost negligible size overhead. SHP is the runner-up method due to its robustness in all three performance factors.

**Special cases.** For distance queries on unweighted complex networks, HL techniques with fast label construction (e.g., SHP+BP, BHP+BP and DHP+BP) exhibit the best all-round performance. For distance queries on undirected road networks, PHL is the most recommended solution as it is specifically designed for this special case.

**Hardness in implementation.** Another important criteria of choosing a HL technique is the difficulty of its implementation. We rank the methods according to our effort put in the source codes. Among all implemented methods, DHP and SHP are the easiest ones which require only a few hundred lines of codes while other methods (e.g., BHP and PHL), they normally require a few thousand or more lines with a complex work flow.

**Graph updates.** In this work, we have conducted a comprehensive experimental study of HL techniques on static graphs. However, graphs in emerging applications may be updated over time. One solution is to incrementally maintain the index [7, 33] subject to the graph changes. However, we do not recommend to maintain the index incrementally as a small portion of updates ($\sim 10\%$) may be already longer than the time of index re-construction. To handle dynamic updates on the graph, we follow Delling et al. [17] and recommend to periodically rebuild the index. SHP is the preferable solution for periodic re-construction due to its lightweight construction process.

| Graph | | Name | Indexing Time ($s$) | | | | | | | Index Size ($MB$) | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | HL | | | Competitors | | | | HL | | | Competitors | | | |
| | | | SHP | SHP +BP | SHP +HLC | PHL | IS-L | | | SHP | SHP +BP | SHP +HLC | PHL | IS-L | | |
| Complex | Directed | GNUTELLA | 18.5 | **18.4** | 28.8 | - | 1723 | | | 184 | 285 | **77** | - | 536 | | |
| | | SLASHDOT | 31.2 | **3.3** | 44.7 | - | 7877 | | | 236 | 181 | **127** | - | 3572 | | |
| | | NOTREDAME | **1.94** | 3.4 | 4.3 | - | 1109 | | | 80 | 685 | **24.9** | - | 561 | | |
| | | DBLP | 4882 | **2506** | 6226 | - | - | | | 17300 | 13221 | **7383** | - | - | | |
| | | WIKI-POLISH | 2760 | **1328** | 3543 | - | 8951 | | | 8740 | 8034 | 4592 | - | **1199** | | |
| | | TRECTWT10G | 61.8 | 77 | 102 | - | **53.2** | | | 1496 | 4378 | **138** | - | 141 | | |
| | | WIKITALK | 212 | **36.6** | 263 | - | 115 | | | 2331 | 4834 | 466 | - | **120** | | |
| | | FLICKR | 3839 | **1955** | 5047 | - | 5339 | | | 17351 | 15186 | 5592 | - | **915** | | |
| | | INDOCHINA | 1214 | **966** | 1465 | - | - | | | 12378 | 26180 | **2016** | - | - | | |
| | | UK2002 | **22322** | - | - | - | - | | | **169750** | - | - | - | - | | |
| | Undir. | CATDOG | 383 | **183** | 458 | 4119 | 6369 | | | 824 | 946 | 561 | 1422 | **404** | | |
| | | COM-DBLP | 162 | **66.9** | 223 | 1572 | 5173 | | | 1121 | 789 | **379** | 1743 | 43169 | | |
| | | SKITTER | 874 | **261** | 1111 | 8811 | 1153 | | | 4188 | 3060 | 1631 | 6773 | **445** | | |
| | | YOUTUBE | 761 | **278** | 784 | 3820 | 2809 | | | 4311 | 4736 | 1361 | 4212 | **845** | | |
| | | HOLLYWOOD | 15648 | **7566** | 19090 | - | - | | | 20961 | 16548 | **10035** | - | - | | |
| Road | | | SHP | N/A | SHP +HLC | PHL | CH | TNR | AH | SHP | N/A | SHP +HLC | PHL | CH | TNR | AH |
| | Undirected | BAY | **4.6** | - | 10.4 | 5.3 | 16.5 | 54.6 | 166 | 177 | - | 24.2 | 116 | **15.6** | 561 | 52.6 |
| | | FLA | **22.4** | - | 46.9 | 29.3 | 37.2 | 109 | 503 | 710 | - | 73 | 521 | **51.5** | 296 | 166 |
| | | CAL | **44.8** | - | 90.7 | 60.3 | 65 | 243 | 787 | 1349 | - | 143 | 1045 | **91.5** | 1154 | 307 |
| | | E | **128** | - | 249 | 169 | 147 | 482 | 1826 | 3360 | - | 275 | 2469 | **171** | 902 | 565 |
| | | W | **212** | - | 420 | 327 | 225 | 885 | 2724 | 5550 | - | 565 | 4603 | **300** | 2227 | 994 |
| | | CTR | 978 | - | 1801 | 1070 | **729** | 2680 | 7291 | 16902 | - | 1114 | 12288 | **671** | 4048 | 2179 |
| | | USA | 1432 | - | 2710 | 1725 | **1049** | 3685 | 12808 | 29548 | - | 1871 | 22528 | **1142** | 3149 | 3747 |

**Table 3: Indexing time and index size**

| Graph | | Name | Response Time ($\mu s$) | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | HL | | | Competitors | | | |
| | | | SHP | SHP +BP | SHP +HLC | PHL | IS-L | | |
| Complex (Distance) | Directed | GNUT. | **2.83** | 3.67 | 9.96 | - | 27.7 | | |
| | | SLAS. | 1.84 | **0.47** | 4.86 | - | 68.4 | | |
| | | NOTR. | **0.38** | 0.47 | 0.99 | - | 7.3 | | |
| | | DBLP | 8.31 | **7.6** | 40.67 | - | - | | |
| | | WIKI-P. | 4.12 | **3.62** | 15.2 | - | 8484 | | |
| | | TRECT. | **1.15** | 1.52 | 13.4 | - | 1182 | | |
| | | WIKI. | 1.45 | **0.47** | 3.13 | - | 202 | | |
| | | FLICKR | 5.46 | **5.07** | 19.3 | - | 24116 | | |
| | | INDO. | **1.54** | 2.09 | 4.03 | - | - | | |
| | | UK2002 | **6.54** | - | - | - | - | | |
| | Undir. | CATDOG | 1.73 | **1.36** | 5.13 | 2.3 | 13942 | | |
| | | COM. | 4.08 | **3.15** | 19.6 | 6.3 | 491 | | |
| | | SKIT. | 2.97 | **1.82** | 12.21 | 4.13 | 922.8 | | |
| | | YOUT. | 1.78 | **1.3** | 5.93 | 2.2 | 644.2 | | |
| | | HOLLY. | **21.1** | **21.1** | 71.3 | - | - | | |
| Road (Distance) | | | SHP | N/A | SHP +HLC | PHL | CH | TNR | AH |
| | Undirected | BAY | 0.53 | - | 3.23 | **0.4** | 31.5 | 1.84 | 21 |
| | | FLA | 0.66 | - | 4.12 | **0.58** | 33.6 | 2.4 | 29 |
| | | CAL | 0.74 | - | 4.86 | **0.63** | 36.7 | 2.25 | 30 |
| | | E | 0.93 | - | 6.12 | **0.71** | 75.1 | 3.8 | 59 |
| | | W | 1.11 | - | 6.3 | **0.71** | 60.6 | 2.8 | 47.5 |
| | | CTR | 1.14 | - | 8.2 | **0.86** | 104 | 3.7 | 71 |
| | | USA | 1.2 | - | 8.74 | **0.89** | 142 | 5.15 | 79 |
| Road (Path) | Undirected | BAY | **19.6** | - | - | 28.3 | 70.4 | 204 | 53 |
| | | FLA | **42.9** | - | - | 82.7 | 122 | 619 | 102 |
| | | CAL | **74.4** | - | - | 157 | 124 | 674 | 109 |
| | | E | **125** | - | - | 195 | 334 | 1538 | 241 |
| | | W | **220** | - | - | 384 | 358 | 1861 | 274 |
| | | CTR | 446 | - | - | 596 | 473 | 2736 | **311** |
| | | USA | 626 | - | - | 805 | 716 | 3671 | **407** |

**Table 4: Response time**

## Acknowledgments

This work was partially supported by MYRG-2016-00182-FST from the UMAC RC, 61502548 from NSFC, GRF 152196/16E from the Hong Kong RGC.

| **Distance Queries** | |
|---|---|
| **Type** | **Top-3 list** |
| Complex | $\text{SHP}_{61\%} \prec \text{SHP+HLCM}_{29\%} \prec \text{BHP}_{6\%}$ |
| Road | $\text{SHP}_{48\%} \prec \text{SHP+HLC}_{30\%} \prec \text{CH}_{16\%}$ |
| **Path Queries** | |
| **Type** | **Top-3 list** |
| Road | $\text{CH}_{83\%} \prec \text{SHP}_{15\%} \prec \text{BHP}_{1\%}$ |

**Table 5: Top-3 methods by preference probability**

# 8. REFERENCES

[1] `http://degroup.cis.umac.mo/sspexp`.

[2] I. Abraham, D. Delling, A. V. Goldberg, and R. F. Werneck. A hub-based labeling algorithm for shortest paths in road networks. In *SEA*, pages 230–241. 2011.

[3] I. Abraham, D. Delling, A. V. Goldberg, and R. F. Werneck. Hierarchical hub labelings for shortest paths. In *ESA*, pages 24–35. 2012.

[4] I. Abraham, A. Fiat, A. V. Goldberg, and R. F. Werneck. Highway dimension, shortest paths, and provably efficient algorithms. In *SODA*, pages 782–793, 2010.

[5] T. Akiba, Y. Iwata, K.-i. Kawarabayashi, and Y. Kawata. Fast shortest-path distance queries on road networks by pruned highway labeling. In *ALENEX*, pages 147–154, 2014.

[6] T. Akiba, Y. Iwata, and Y. Yoshida. Fast exact shortest-path distance queries on large networks by pruned landmark labeling. In *SIGMOD*, pages 349–360, 2013.

[7] T. Akiba, Y. Iwata, and Y. Yoshida. Dynamic and historical shortest-path distance queries on large evolving networks by pruned landmark labeling. In *WWW*, pages 237–248, 2014.

[8] M. Babenko, A. V. Goldberg, H. Kaplan, R. Savchenko, and M. Weller. On the complexity of

hub labeling. In *MFCS*, pages 62–74, 2015.

[9] H. Bast, D. Delling, A. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner, and R. F. Werneck. Route planning in transportation networks. *arXiv preprint arXiv:1504.05140*, 2015.

[10] H. Bast, S. Funke, P. Sanders, and D. Schultes. Fast routing in road networks with transit nodes. *Science*, 316(5824):566–566, 2007.

[11] R. Bauer and D. Delling. Sharc: Fast and robust unidirectional routing. *JEA*, 14:4, 2009.

[12] R. Bauer, D. Delling, P. Sanders, D. Schieferdecker, D. Schultes, and D. Wagner. Combining hierarchical and goal-directed speed-up techniques for dijkstra's algorithm. *JEA*, 15:2–3, 2010.

[13] P. Boldi and S. Vigna. The WebGraph framework I: Compression techniques. In *WWW*, 2004.

[14] U. Brandes and C. Pich. Centrality estimation in large networks. *International Journal of Bifurcation and Chaos*, 17(07):2303–2318, 2007.

[15] E. Cohen, E. Halperin, H. Kaplan, and U. Zwick. Reachability and distance queries via 2-hop labels. *SICOMP*, 32(5):1338–1355, 2003.

[16] D. Delling, A. V. Goldberg, T. Pajor, and R. F. Werneck. Robust exact distance queries on massive networks. *Microsoft Research, USA, Tech. Rep*, 2, 2014.

[17] D. Delling, A. V. Goldberg, T. Pajor, and R. F. Werneck. Customizable route planning in road networks. *Transportation Science*, 2015.

[18] D. Delling, A. V. Goldberg, and R. F. Werneck. Hub label compression. In *SEA*, pages 18–29. 2013.

[19] C. Demetrescu, A. Goldberg, and D. Johnson. 9th dimacs implementation challenge–shortest paths. *American Mathematical Society*, 2006.

[20] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1(1):269–271, Dec. 1959.

[21] A. W.-C. Fu, H. Wu, J. Cheng, and R. C.-W. Wong. Is-label: an independent-set based labeling scheme for point-to-point distance querying. *PVLDB*, 6(6):457–468, 2013.

[22] R. Geisberger, P. Sanders, and D. Schultes. Better approximation of betweenness centrality. In *ALENEX*, pages 90–100, 2008.

[23] R. Geisberger, P. Sanders, D. Schultes, and D. Delling. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *WEA*, pages 319–333. 2008.

[24] A. V. Goldberg and C. Harrelson. Computing the shortest path: A search meets graph theory. In *SODA*, pages 156–165, 2005.

[25] A. V. Goldberg, H. Kaplan, and R. F. Werneck. Reach for a*: Shortest path algorithms with preprocessing. In *The Shortest Path Problem*, pages 93–140, 2006.

[26] R. J. Gutman. Reach-based routing: A new approach to shortest path algorithms optimized for road networks. *ALENEX/ANALC*, 4:100–111, 2004.

[27] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[28] M. Jiang, A. W.-C. Fu, R. C.-W. Wong, and Y. Xu. Hop doubling label indexing for point-to-point distance querying on scale-free networks. *PVLDB*, 7(12):1203–1214, 2014.

[29] R. Jin, N. Ruan, Y. Xiang, and V. Lee. A highway-centric labeling approach for answering distance queries on large sparse graphs. In *SIGMOD*, pages 445–456, 2012.

[30] J. Kunegis. Konect: the koblenz network collection. In *WWW*, pages 1343–1350, 2013.

[31] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[32] M. Potamias, F. Bonchi, C. Castillo, and A. Gionis. Fast shortest path distance estimation in large networks. In *CIKM*, pages 867–876, 2009.

[33] Y. Qin, Q. Z. Sheng, and W. E. Zhang. Sief: Efficiently answering distance queries for failure prone graphs. In *EDBT*, pages 145–156, 2015.

[34] P. Sanders and D. Schultes. Engineering highway hierarchies. In *ESA*, pages 804–816, 2006.

[35] M. Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *JACM*, 51(6):993–1024, 2004.

[36] M. V. Vieira, B. M. Fonseca, R. Damazio, P. B. Golgher, D. d. C. Reis, and B. Ribeiro-Neto. Efficient search ranking in social networks. In *CIKM*, pages 563–572, 2007.

[37] F. Wei. Tedi: efficient shortest path query answering on graphs. In *SIGMOD*, pages 99–110, 2010.

[38] L. Wu, X. Xiao, D. Deng, G. Cong, A. D. Zhu, and S. Zhou. Shortest path and distance queries on road networks: An experimental evaluation. *PVLDB*, 5(5):406–417, 2012.

[39] Y. Yano, T. Akiba, Y. Iwata, and Y. Yoshida. Fast and scalable reachability queries on graphs by pruned labeling with landmarks and paths. In *CIKM*, pages 1601–1606, 2013.

[40] A. D. Zhu, H. Ma, X. Xiao, S. Luo, Y. Tang, and S. Zhou. Shortest path and distance queries on road networks: towards bridging theory and practice. In *SIGMOD*, pages 857–868, 2013.