Landmark Selection Strategy to Accelerate Shortest Distance Queries

ZHIPENG HE, Guangzhou University, China
DIAN OUYANG*, Guangzhou University, China
JIANYE YANG, Guangzhou University, China

Computing the shortest path between vertices is a fundamental task in network analysis. This paper introduces an innovative strategy for selecting landmarks in a graph, which improves upon the traditional highest-degree selection method. Our approach captures the essential structural features of graphs, effectively reducing landmark overlap and clustering. Combined with partition querying and upper-bound pruning optimizations, our algorithm achieves a notable $20\% - 30\%$ increase in query speed and is applicable to multi-million record datasets. This study offers new insights and solutions for enhancing query performance in large-scale graph-structured data, paving the way for further innovations in graph database optimization to meet the growing demands of large-scale graph data processing.

CCS Concepts: • **Theory of computation → Graph algorithms analysis**.

Additional Key Words and Phrases: Shortest path, landmark, distance labeling, optimization

1 INTRODUCTION

Finding the shortest path is a core problem in data mining and network analysis when dealing with large-scale graph data, with a wide range of applications including but not limited to traffic planning, social network analysis, and network routing optimization[9]. With the continuous growth of data scale, traditional shortest path algorithms face enormous challenges in terms of efficiency and scalability. How to quickly and accurately calculate the shortest path has become an urgent problem that needs to be solved.

Current state-of-the-art algorithms in shortest path computations leverage landmark labeling-based approaches, which typically require specifying a parameter 'K' to denote the number of landmarks [2, 6]. These algorithms conventionally select the top 'K' vertices with the highest degree values within the graph to serve as landmarks.

1.1 Motivation

We believe that there is still optimization space by using only the top k vertices with the highest degree as landmarks. To try different strategies, we considered using the top k vertices with the highest betweenness centrality as the landmark. However, although there has been some improvement in label size, it has not shown significant improvement in actual queries. More importantly, due to the high time-consuming burden of calculating betweenness centrality in large graphs, we have to abandon this method. By analyzing the strategy of directly using vertices with the highest degree and betweenness centrality as landmarks [2, 6, 7], we found two main issues: firstly, the landmarks selected based on betweenness centrality overlap with those selected by the degree, and this situation is particularly prominent in large networks. Secondly, in highly clustered networks, strong clustering phenomena may lead to a core–periphery structure [8, 11], resulting in relatively clustered selected vertices that we expect to be more dispersed in order to better cover the entire graph. Therefore, we are trying to find new algorithmic inspirations.

*Corresponding author.

1.2 Our idea

We propose a novel landmark selection strategy: (1) Firstly, we use the Top-k degree vertices as benchmark landmarks and assign other vertices to the region where their nearest landmarks are located. (2) Next, within each region, we further select the vertex with the highest degree as the new landmark. This process aims to address vertex overlap and clustering issues, thereby improving query efficiency. It should be emphasized that in the selection of new landmarks, we only focus on the vertices with the highest degree within each region, without considering vertices across regions. This detail ensures that the new landmarks can be scattered throughout the entire graph and not clustered in a specific local area. (3) During the query phase, we further introduced optimization for partitioned queries.

1.3 Contribution

In this paper, our main contributions are as follows:

- An innovative landmark selection strategy in the graph has been proposed, which can better capture important features of the graph structure and effectively reduce the overlap and clustering phenomena between landmarks compared to the traditional highest degree selection method.
- The concept of region partitioning has been introduced, assigning vertices to the regions where their nearest landmarks are located, resulting in a more dispersed distribution of landmarks throughout the entire graph, reducing clustering and improving query efficiency.
- The proposed landmark selection strategy and query optimization method have good scalability and universality, and are suitable for various practical application scenarios such as graph databases and social networks. Even when processing billion level datasets, they can perform well, providing strong support for research and applications in related fields.

1.4 Outline

The paper is structured as follows. In Section 2, we introduce the problem definitions and preliminary. We briefly describe the existing solutions in Section 3. In Section 4, we introduce the Distance Query Framework, while Section 5 is dedicated to optimizing the landmark selection strategy. Section 6 presents experiments that evaluate our methods. Finally, Section 7 concludes this paper.

2 PRELIMINARY

Let $G = (V, E)$ be a graph, where V is the set of vertices and E is the set of edges. We denote the number of vertices and edges by $n = |V|$ and $m = |E|$, respectively. For simplicity, we assume that the G is connected and undirected throughout this paper, since our work can be easily extended to directed or disconnected graphs. Let $G[V \setminus R]$ represent removing all landmarks set in R from G, and $d_G(s, t)$ represent the shortest distance from s to t.

2.1 Distance labeling

Let $R \subseteq V$ be a set of special vertices in G, called landmarks, a label $L(v)$ for each vertex $v \in V$ can be precomputed, which can be answered in linear-time by looking up the label $L(v)$. The collection of labels $L = \{L(v)\}_{v \in V}$ is referred to as a distance labeling on graph G. The distance between two vertices s and t can be calculated as:

$$\mathrm{d}_G(s, t) = \min_{r \in L(s) \cap L(t)} (\mathrm{d}_G(s, r) + \mathrm{d}_G(r, t))$$

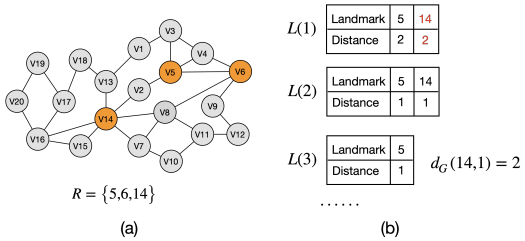


Fig. 1. An explanation of distance labeling:(a) an example graph G, (b) Compute a label $L(v)$ for each vertex $v \in V$ in G in advance.

Example 1. *Consider the graph G depicted in Fig.1(a), with a set of landmarks* $R = \{5, 6, 14\}$ *and precomputed distance labels* $L(1) = \{(5, 2), (14, 2)\}$, $L(2) = \{(5, 1), (14, 1)\}$ *and* $L(3) = \{(5, 1)\}$, *etc. in Fig.1(b), we can efficiently determine that* $d_G(14, 1) = 2$.

2.2 Problem Definition

Given a graph G and a set of precomputed landmarks $R \subseteq V$, a shortest distance query $q = (s, t)$ is defined, where $s, t \in V$, and the answer is the shortest distance $d_G(s, t)$. This paper aims to develop efficient landmark selection strategies to optimize query performance. Specifically, our objective is to minimize the total query time, $T(Q) = min\sum_{q \in Q} T(q, R)$ where $T(q, R)$ represents the time taken to answer the query q using the set of landmarks R.

Example 2. *Fig.1(a) shows a network* $G = (V, E)$ *with 20 vertices and 30 edges. There are many paths between* v_{12} *and* v_{19}. *For example, two of them are* $p_1 = (v_{12}, v_{11}, v_8, v_{14}, v_{13}, v_{18}, v_{17}, v_{19})$ *and* $p_2 = (v_{12}, v_9, v_6, v_5, v_2, v_{14}, v_{16}, v_{20}, v_{19})$ *with* $d_G(p_1) = 7$ *and* $d_G(p_2) = 8$. p_1 *is a shortest path between* v_{12} *and* v_{19} *as covered by landmark* v_{14}. *Given a shortest distance query* $q = (v_{12}, v_{19})$, *the answer of* q *is* $d(v_{12}, v_{19}) = 7$.

3 EXISTING SOLUTIONS

This section formalizes two distance labeling concepts: 2-hop labeling and Highway cover labeling[4, 6], which lay the foundation for our theoretical analysis and new algorithm.

3.1 2-Hop Labeling

The 2-hop labeling technique is commonly used in methods for shortest path computation[1, 2, 5, 6, 9]. Given a network G, 2-hop labeling assigns each vertex $v \in V$ a label $L(v)$ which is a collection of pairs $(u, \mathrm{d}_G(v, u))$ where $u \in V$.

Definition 1. ***(2-hop cover labeling)***. *A labeling L is a 2-hop Cover labeling of G if, for each pair* $s, t \in V$, $L(s) \cap L(t)$ *contains at least one common vertex r on the shortest path from s to t. This guarantees that:*

$$Query_(s, t, L) = \min\{\delta_L(r, s) + \delta_L(r, t) \mid r \in L(s) \cap L(t)\}$$

Thus, we determine $d_G(s, t)$ by scanning both $L(s)$ and $L(t)$, with a query time complexity of $O(|L(s)| + |L(t)|)$.

3.2 Highway Cover Labeling

We start by providing the definitions of highway and highway cover labeling scheme [3, 6].

Definition 2. ***(Highway)***. *A highway H can be defined as a tuple* (R, δ_H) *consists of a distance decoding function* δ_H *and a set R of landmarks, the value* δ_H *corresponds to the shortest path distance* $d_G(r_1, r_2)$, *i.e.* $\delta_H(r_1, r_2) = d_G(r_1, r_2)$.

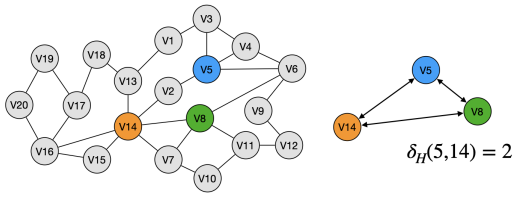
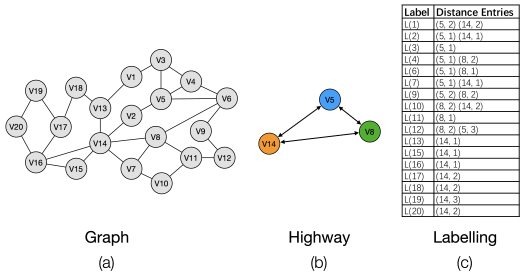


Fig. 2. Highway Labeling



Label	Distance Entries
L(1)	(5, 2) (14, 2)
L(2)	(5, 1) (14, 1)
L(3)	(5, 1)
L(4)	(5, 1) (8, 2)
L(6)	(5, 1) (8, 1)
L(7)	(5, 1) (14, 1)
L(9)	(5, 2) (8, 2)
L(10)	(8, 2) (14, 2)
L(11)	(8, 1)
L(12)	(8, 2) (5, 3)
L(13)	(14, 1)
L(15)	(14, 1)
L(16)	(14, 1)
L(17)	(14, 2)
L(18)	(14, 2)
L(19)	(14, 3)
L(20)	(14, 2)

Fig. 3. Highway Cover Labeling

Example 3. *Refer to the graph G shown in the Fig. 2, we select the set $R = \{5, 8, 14\}$ as landmarks. The distance decoding function δ_H is specified as follows: $\delta_H(5, 14)$ denotes the shortest path distance from vertices 5 to 14, which is 2. Similarly, $\delta_H(5, 8)$ is 2 and $\delta_H(8, 14)$ is 1.*

This highway H effectively represents the landmark distance relationships within graph G, facilitating the rapid computation or estimation of shortest paths between vertices.

Definition 3. ***(Highway Cover Labeling)**. A highway cover labelling $\Gamma = (H, L)$ consists of a highway H and a distance labeling L. Then for any vertices $u \in V \backslash R$ and for any $r \in R$, there exist $r' \in R$ in $L(u)$ such that r' lies on the shortest path from u to r (where r and r' may be the same).*

Example 4. *In the network illustrated in the Fig. 3(a), the highway H includes three landmarks:* 5, 8, *and* 14, *highlighted in Fig. 3(b). The path $p_1 = (v_3, v_5, v_4)$ denotes the shortest path between vertices* 3 *and* 4, *which is constrained by landmark* 5. *In contrast, neither of paths $p_2 = (v_3, v_1, v_{13}, v_{14}, v_2, v_5, v_4)$ and $p_3 = (v_3, v_4)$ satisfies the condition of being a shortest path between 3 and 4 constrained by landmark 5.*

In contrast to a 2-hop cover labeling, a highway cover labeling can only answer distance queries between landmarks and other vertices in the graph, making it a partial distance labeling[5].

How to choose a better landmark set will be explained in detail in the section 5, which is the main part of our algorithm.

4 DISTANCE QUERY FRAMEWORK

In this section, we present the distance query framework, which facilitates efficient and accurate batch computation of shortest distances for any pair of vertices in a large graph.

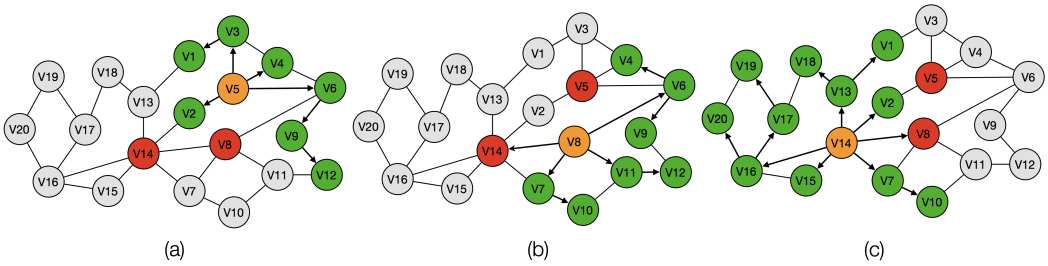


Fig. 4. Label Construction.

4.1 Label Construction

Algorithm 1: Construct highway cover labeling L

Input: Network $G(V, E)$, Highway $H(R, \delta_H)$
Output: The highway cover labeling L
$L(v) \leftarrow \emptyset$ for all $v \in V \setminus R$;
foreach $r_i \in R$ **do**
  $Q_L \leftarrow \emptyset$;
  $Q_N \leftarrow \emptyset$;
  Q_L.push(r_i);
  $n \leftarrow 0$;
  while Q_L *and* Q_N *are not empty* **do**
    foreach $u \in Q_L$ *at depth* n **do**
      foreach $v \in adj[u]$ **do**
        if v *is unvisited* **then**
          $\delta_{\text{BFS}}(r_i, v) \leftarrow n + 1$;
          if v *is a landmark* **then**
            Q_N.push(v);
          else
            Q_L.push(v);
            $L(v) \leftarrow L(v) \cup \{(r_i, \delta_{\text{BFS}}(r_i, v))\}$;
    foreach $u \in Q_N$ *at depth* n **do**
      foreach $v \in adj[u]$ **do**
        if v *is unvisited* **then**
          $\delta_{\text{BFS}}(r_i, v) \leftarrow n + 1$;
          Q_N.push(v);
    $n \leftarrow n + 1$;
return L

The label construction in our algorithm is outlined in Algorithm 1. The procedure begins by performing a Breadth-First Search(BFS) from each landmark $r \in R$. During this process, a distance entry $(r, \delta(r, v))$ is added to the label of a vertex $v \in V \setminus R$ if no closer landmark $r' \in R$ lies on the shortest path from r to v. This condition ensures that the

label for v reflects the shortest distance from the most proximal landmark, optimizing the search process in our graph queries.

In Fig. 4, the green vertices are visited and stored, while the grey vertices are pruned and no stored. This is because when BFS traversal begins at landmark r, grey vertices are obstructed by other landmarks r'.

4.2 Query Distance

For any pair of vertices s and t, a highway cover distance labeling scheme L can be used to estimate an upper bound $\underline{d}_{s,t}$ for the shortest path distance from s to t can be determined as follows,

$$\underline{d}_{s,t} = distance[s][i] + distance[t][i] \tag{1}$$

$$\underline{d}_{s,t} = distance[s][i] + highway[i][j] + distance[t][j] \tag{2}$$

Here, Equation (1) represents the shortest distance from s to t passing through the landmark r_i. Equation (2) represents the shortest distance from s to t passing through both landmarks r_i and r_j, where the distance between r_i and r_j can be directly obtained from the highway.

We conduct distance query as described in Algorithm 2. When s and t are in the same region, the upper bound is calculated using Equation (1). Conversely, when s and t are in different regions, the upper bound is computed using Equation (2). The next section we will provide a detailed explanation of how the regions are partitioned.

Algorithm 2: Query Distance

```
Input: s, t
Output: d_G(s, t)
if vertexToRegion[s] == vertexToRegion[t] then
    Initialize queues and distances;
    dist_upper ← landmark_distances[s][region] + landmark_distances[t][region];
    while both queues are not empty do
        Select the smaller queue;
        while selected queue is not empty do
            foreach w in neighbors of current node do
                if w already visited by other queue then
                    return path length via w;
                Mark w and push to queue;
    Reset and clear queues;
    return min(res, dist_upper);
else
    Initialize m ← 99;
    foreach i in C[s] do
        foreach j in C[t] do
            m ← min(m, distances[s][i] + highway[vertices[s][i]][vertices[t][j]] + distances[t][j]);
    return m;
```

Next, we introduce hash region and partition query for further optimization. In our algorithm, hash region technique is a kind of hash table to quickly determine if vertices s and t belong to the same region. During the query process, we

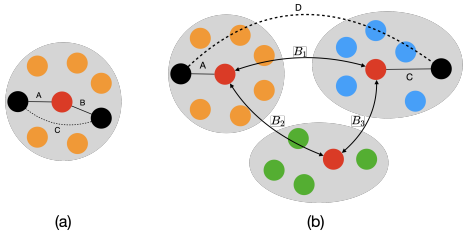


Fig. 5. Partition Query

first Check if s and t are in the same region using the hash region. If they are, distance[s][i] + distance[t][i] is used as the upper bound pruning condition. If not, distance[s][i] + highway[i][j] + distance[t][j] serves as the upper bound. Subsequently, we conduct a simultaneous bidirectional search initiated from both s and t in $G[V \setminus R]$ under the upper bound $\underline{d}_{s,t}$.

Example 5. *Refer to the graph shown in Fig. 5, black vertices represent the vertices to be queried, and red vertices represent the finalized landmarks. In Fig. 5(a), after region-based hash determines that s and t belong to the same region, a bidirectional BFS is executed. The process terminates when the distance C exceeds A + B, at which* $d_G(s,t) = A + B$ *is confirmed as the shortest distance. In Fig. 5(b), orange, green, and blue vertices each represent vertices within the same region. After region-based hash determines that s and t do not belong to the same region, a bidirectional BFS is executed. The process terminates once the distance D exceeds* $A + \min\{B1, B2 + B3\} + C$*, establishing* $d_G(s,t) = A + \min\{B1, B2 + B3\} + C$ *as the shortest distance. Note that,* B_1*,* B_2 *and* B_3 *can be directly obtained through the highway.*

5 LANDMARK SELECTION

In this section, we propose an innovative strategy for selecting important vertices, referred to as landmarks, in a graph. This approach seeks to improve upon the traditional method of selecting the top-K vertices with the highest degrees by incorporating a more nuanced iterative regional division process.

Algorithm 3 describes how our algorithm select landmarks. The strategy enhances efficiency and effectiveness in graph processing tasks. The detailed methodology is as follows: (i) **Initial Step:** We begin by selecting the top-K vertices with the highest degrees in the graph as the initial set of landmarks. This step serves as the foundation for the subsequent iterative process, providing initial vertices for the refinement of landmark selection. (ii) **Regional Division Step:** Next, each vertex not already designated as a landmark is assigned to the region associated with the nearest landmark. This division creates a dedicated influence zone for each landmark, simplifying the search space for subsequent queries and thus enhancing query speed. (iii) **New Landmark Selection Step**: Each region is considered independently, which only considers the connections within the region and ignores the connections across regions. On this basis, we select a vertex with the highest degree within each region as a new landmark to ensure efficiency and accuracy in query operations.

Through such strategy, we construct a set of landmarks that not only considers vertex significance but also optimizes spatial locality, significantly speeding up subsequent algorithm queries. The essence of this method lies in its ability to rapidly locate the area relevant to a query, reducing unnecessary search and computation, offering a new pathway for efficient processing of large-scale graph data.

Algorithm 3: Landmark Selection

Input: $G = (V, E)$;
Output: $topk[]$;
Function `SelectLandmarks(`*Graph* G`)`**:**
  $topk[] \leftarrow$ select K highest degree vertices as landmarks;
  foreach *landmark* j *in* $topk$ **do**
    Perform BFS from j to calculate distances;
  end
  foreach $i = 0$ **to** $V - 1$ **do**
    if i *not in* $topk$ **then**
      Find the nearest landmark for vertex i and assign to its region
    end
  end
  foreach $i = 0$ **to** $K - 1$ **do**
    foreach *vertex in* $regions[i]$ **do**
      Find MaxDgreeVertex;
    end
    $topK[vertex] \leftarrow MaxDgreeVertex$
  end
  return $topk[]$;

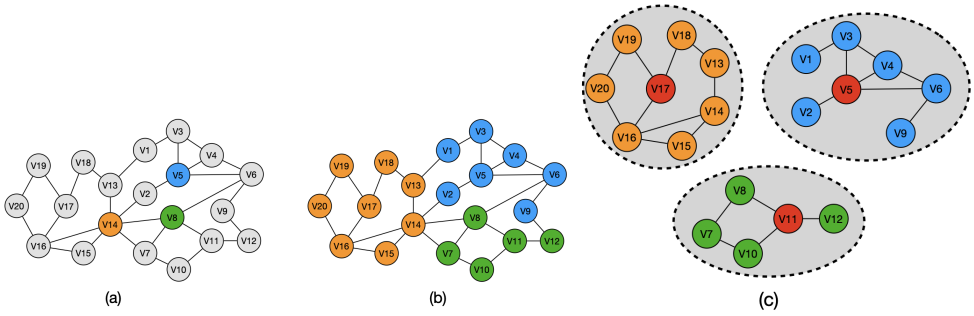


Fig. 6. Landmark Selection.

It should be noted that due to the Order Independence [6] property of highway cover labeling, there is no need to consider the attribution of a vertex to different landmarks at the same distance. The vertices that are first traversed belong to the region covered by that landmark.

Proof. Defind $OrderL_1$ and $OrderL_2$ as two distance labeling orders over landmark R. During the label construction process, the BFS conducted from landmark r in $OrderL_1$ and $OrderL_2$ traverses the same set of vertices, resulting the same pruded BFS tree. Thus, Algorithm 1 ensures that $L_1(v) = L_2(v)$ for every $v \in V \backslash R$. □

Example 6. *In the network illustrated in Fig. 6(a), vertices* $\{v_5, v_8, v_{14}\}$*, which have the highest degrees, are selected as initial landmarks. Starting from each landmark, recording the vertices reachable by each landmark, as shown in Fig. 6(b). Subsequently, as detailed in Fig. 6(c), each region is treated independently, with marginalization of vertices, meaning only*

intra-region edges are counted while inter-region connections are excluded. Within each region, the vertex with the highest degree, vertices $\{v_5, v_{11}, v_{17}\}$ *marked in red, is selected as the final landmark.*

6 EXPERIMENT

In this section, we present extensive experiments to evaluate our methods. All algorithms were implemented in C++11 using the Standard Template Library (STL) and tested on a server with an Intel Xeon 2.10GHz CPU, 256GB of RAM, and Ubuntu 20.04. The experimental code was written in C++.

6.1 Datasets

Table 1. Datasets

Network	**Vertices** n	**Edges** m
Douban	154,909	654,324
Amazon	334,863	1,851,744
DBLP	317,980	2,099,732
Youtube	1,134,890	5,975,248
As-skitter	1,696,415	22,190,596

We use five publicly available real-world networks from the Stanford Network Analysis Project[10]. Table 1 provides details of these datasets, which we treated as undirected graphs.

6.2 Query Performance Comparison

Table 2. Random vertex pair shortest distance batch query

Datasets	HL	Naive-KHL	KHL
douban	0.806s	0.799s	0.613s
amazon	5.863s	5.841s	4.201s
dblp	1.96s	1.791s	1.538s
youtube	6.584s	5.863s	5.164s
as-skitter	11.816s	11.174s	8.933s

To evaluate the average query time, we randomly generate one million vertex pairs from each graph for batch processing of shortest path distance queries. The results are shown in Table 2. Our algorithm KHL denotes a region hash optimization based on the use of a new landmark selection strategy. The Naive-KHL method denotes no region hash optimization. This optimization is described in Section 4.2. The HL algorithm used in our experiments is derived from[6] which selects the top k highest degree vertices as landmarks without any optimization during the query phase. Experimental validation demonstrated that our algorithm enhances query speed by 20%-30% in large networks compared to traditional methods.

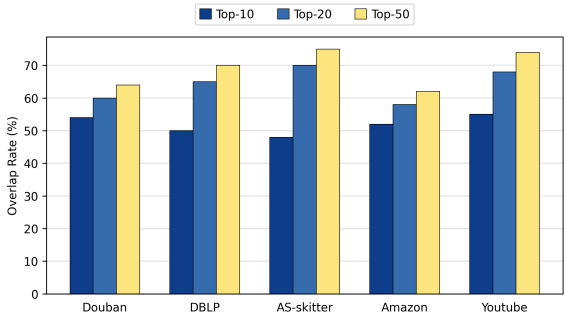


Fig. 7. Overlap rates

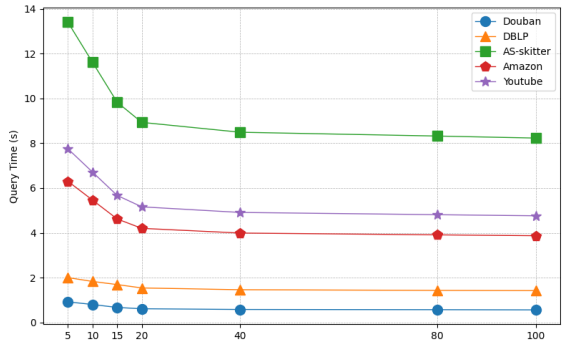


Fig. 8. Performance under Varying Landmarks

6.3 Overlap rates

We conducted experimental analysis on the selected vertices as landmarks using Top-k degree and Top-k betweenness methods. The horizontal axis represents the dataset, and the vertical axis represents the overlap rate. We compared the overlap rates of selected vertices as landmarks under the conditions of setting the number of landmarks at 10, 20, and 50, respectively. Fig. 7 shows that these two traditional methods have a high overlap rates in the selected landmarks. Due to the properties of degree, vertices near those with a higher degree also tend to have higher degrees, resulting in a clustered arrangement of landmarks. The experiments indicated that the high overlap rate also leads to clustering when landmarks are selected based on betweenness. Thus, ensuring a dispersed distribution of landmarks, which can cover more shortest paths between vertex pairs, is a crucial optimization. Our method ensures zero overlap with the landmarks selected by traditional methods.

6.4 Performance under Varying Landmarks

From Fig. 8, it can be seen that as the number of landmarks increases, the query speed also increases accordingly. However, adding additional landmarks will result in an increase in preprocessing time and storage space. When the number of landmarks exceeds 20, the benefits gradually decrease, and further increasing the number of landmarks is difficult to bring effective improvement. Therefore, we consider 20 landmarks to be an optimal choice.

7 CONCLUSION

In this study, we propose and implement an innovative landmark selection strategy that improves traditional degree-based methods, significantly enhancing the effectiveness of landmark selection while reducing redundancy and clustering. By integrating partition query and upper-bound pruning optimizations, our algorithm achieved a notable improvement of 20% to 30% in query speed and can be applied to datasets comprising tens of millions of records. This research provides new insights and solutions for improving query performance in large-scale graph-structured data, such as graph databases and social networks. Moving forward, we will continue to explore more innovative strategies for optimizing graph databases to meet the growing demands of large-scale graph data processing.

8 ACKNOWLEDGMENTS

According to relevant requirements, this section only makes an academic statement to ensure compliance with research standards and originality requirements. The partial content of the article is only to polish academic writing using the artificial intelligence tool (ChatGPT).

REFERENCES

[1] Ittai Abraham, Daniel Delling, Andrew V Goldberg, and Renato F Werneck. 2011. A hub-based labeling algorithm for shortest paths in road networks. In *Experimental Algorithms: 10th International Symposium, SEA 2011, Kolimpari, Chania, Crete, Greece, May 5-7, 2011. Proceedings 10*. Springer, 230–241.

[2] Takuya Akiba, Takanori Hayashi, Nozomi Nori, Yoichi Iwata, and Yuichi Yoshida. 2015. Efficient top-k shortest-path distance queries on large networks by pruned landmark labeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 29.

[3] Takuya Akiba, Yoichi Iwata, Ken-ichi Kawarabayashi, and Yuki Kawata. 2014. Fast shortest-path distance queries on road networks by pruned highway labeling. In *2014 Proceedings of the sixteenth workshop on algorithm engineering and experiments (ALENEX)*. SIAM, 147–154.

[4] Edith Cohen, Eran Halperin, Haim Kaplan, and Uri Zwick. 2003. Reachability and distance queries via 2-hop labels. *SIAM J. Comput.* 32, 5 (2003), 1338–1355.

[5] Muhammad Farhan, Qing Wang, and Henning Koehler. 2022. Batchhl: Answering distance queries on batch-dynamic networks at scale. In *Proceedings of the 2022 International Conference on Management of Data*. 2020–2033.

[6] Muhammad Farhan, Qing Wang, Yu Lin, and Brendan Mckay. 2018. A highly scalable labelling approach for exact distance queries in complex networks. *arXiv preprint arXiv:1812.02363* (2018).

[7] Daoliang He, Pingpeng Yuan, and Hai Jin. 2024. Answering reachability queries with ordered label constraints over labeled graphs. *Frontiers of Computer Science* 18, 1 (2024), 181601.

[8] Wentao Li, Miao Qiao, Lu Qin, Ying Zhang, Lijun Chang, and Xuemin Lin. 2020. Scaling up distance labeling on graphs with core-periphery properties. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1367–1381.

[9] Dian Ouyang, Lu Qin, Lijun Chang, Xuemin Lin, Ying Zhang, and Qing Zhu. 2018. When hierarchy meets 2-hop-labeling: Efficient shortest distance queries on road networks. In *Proceedings of the 2018 International Conference on Management of Data*. 709–724.

[10] Stanford Network Analysis Project. 2024. Stanford Large Network Dataset Collection. https://snap.stanford.edu/data/index.html Accessed: 2024-04-09.

[11] Wikipedia. 2024. Clustering Coefficient. https://en.wikipedia.org/wiki/Clustering_coefficient Accessed: 2024-04-09.