Optimizing Distance Computation in Distributed Graph Systems

QING WANG, SHENGYI JI, PENG PENG\textsuperscript{*}, (Member, IEEE), MINGDAO LI, PING HUANG, AND ZHENG QIN\textsuperscript{**}, (Associate Member, IEEE)

Hunan University, Changsha 410082, China
Corresponding author: Peng Peng (hnu16pp@hnu.edu.cn)

ABSTRACT
Given a large graph, such as a social network or a knowledge graph, a fundamental query is how to find the distance from a source vertex to another vertex in the graph. As real graphs become very large and many distributed graph systems, such as Pregel, Pregel+, Giraph, and GraphX, are proposed, how to employ distributed graph systems to process single-source distance queries should attract more attention. In this paper, we propose a landmark-based framework to optimize the distance computation over distributed graph systems. We also use a measure called \textit{set betweenness} to select the optimal set of landmarks for distance computation. Although we can prove that selecting the optimal set of landmarks is NP-hard, we propose a heuristic distributed algorithm that can guarantee the approximation ratio. Experiments on large real graphs confirm the superiority of our methods.

INDEX TERMS
Distributed information systems, graph theory, distributed computing, distance computation, distributed graph systems, landmark.

I. INTRODUCTION
With the rapid development of the Internet and social networks in recent years, large-scale data in graph models have gradually increased. As a general data model, the graph model is widely used in many applications. For example, the Internet can be represented as a graph. In the graph, each web page is a vertex, and a hyperlink between the web pages acts as an edge; in a social network, each person is a vertex of the graph, and the relationship between persons constitutes the edge. In addition, many information systems, such as search engines and recommender systems, also use graphs as information carrier models.

When graph models are used in an increasing number of applications, as one of the most classic problems in a graph, single-source shortest path length (SSSP length) queries have been studied for more than half a century and have received increasing attention [6], [16], [26]. Given a graph and a source vertex, an SSSP length query finds the distance from the source vertex to each vertex in the graph. The SSSP length query is widespread in many real applications. For example, the selection of travel routes in traffic networks is one of the most basic applications of SSSP length. In addition, the transmission efficiency of information flow in a computer network is optimal when it is transmitted via the shortest router sequences.

As the sizes of real graphs increase, the classic solutions for SSSP length queries in real graphs become inefficient. Additionally, the emergence of distributed graph systems, including Pregel [16], Pregel+ [25], Giraph [21] and GraphX [9], is inevitable to maintain large graphs. These distributed graph systems follow the vertex-centric BSP (bulk synchronous parallel) computing model, which divides the calculation into a series of superstep iterations. To improve the scalability of the SSSP length query evaluation, we propose a landmark-based framework over distributed graph systems for computing the SSSP length query in large graphs in this paper.

The computational framework in this paper first selects a sequence of appropriate landmarks based on a measure named \textit{set betweenness}, which are extended from [7], [15] and provides a landmark selection criterion by evaluating the number of shortest paths covered by the set of landmarks.
Although we prove that selecting the optimal set of vertices with the maximal set betweenness as the landmarks is NP-hard, our proposed heuristic distributed algorithm can guarantee the approximation ratio. Then, we take advantage of the calculated shortest path trees of the landmarks to compute the distances from the source vertex of the SSSP length query to other vertices over distributed graph systems.

In summary, the main contributions we make are as follows:

• We propose a landmark-based framework to evaluate the SSSP length query over distributed graph systems, which utilizes the characteristics of the distributed graph systems to improve the efficiency and scalability of our method.

• We formulate the measure named set betweenness to select the optimal set of landmarks. Although we prove that the problem of selecting the optimal set of landmarks is NP-hard, we propose a heuristic distributed strategy to guarantee the approximation ratio.

• Finally, we conduct extensive experiments over different kinds of real graphs on multiple distributed graph systems (Pregel + [25], Giraph [21] and GraphX [9]) to verify the performance of the proposed techniques.

II. PRELIMINARIS

In this section, we introduce related concepts throughout this paper.

A. GRAPH AND PATH

In this paper, we consider an undirected, unweighted graph $G = (V, E)$, where $V$ is a set of vertices and $E \subseteq V \times V$ is a set of edges. Each edge $e = (v, u) \in E$ is an edge connecting two vertices $u$ and $v$. It should be noted that we mainly consider the undirected and unweighted graphs in this paper. However, our method can also be applied to directed and weighted graphs. For example, Figure 1 shows a graph used as a running example in this paper.

Given two vertices $s, t \in V$, we define a sequence $p(s, t) = (s, v_1, v_2, \ldots, v_{t-1}, t)$ to denote a path length between two vertices $s$ and $t$, where $\{v_1, v_2, \ldots, v_{t-1}\} \subseteq V$ and $\{(s, v_1), (v_1, v_2), \ldots, (v_{t-1}, t)\} \subseteq E$. Then, the distance $d(s, t)$ between vertices $s$ and $t$ is defined as the length of the shortest path between $s$ and $t$.

![Figure 1. Example graph.](image)

A shortest path tree rooted at vertex $v$ is a spanning tree $T$ of $G$, such that the distance from $v$ to any other vertex $u$ in $T$ is equal to the distance from $v$ to $u$ in $G$. Here, we denote the shortest path tree rooted at vertex $v$ as $SPT(v)$. Furthermore, for each vertex $v'$ in $SPT(v)$, its parent in $SPT(v)$ is denoted as $v'.p_v$, and its distance to $v$ is denoted as $v'.d_v$. Here, the parent of $v$ in $SPT(v)$ is $v$ itself. For example, Figure 2(a) and Figure 2(b) show $SPT(v_6)$ and $SPT(v_7)$, which are the shortest path trees rooted at $v_6$ and $v_7$ for the example graph. For $v_2$, its parent $v_2.p_{v_6}$ in $SPT(v_6)$ is $v_1$ and $v_2.d_{v_6}$ is 3.

![Figure 2. Example shortest path trees.](image)

In this paper, we study the problem of answering single-source shortest path length (SSSP length) queries. Given a graph $G$ and a source vertex $s$, an SSSP length query finds the distances from $s$ to every vertex in $G$.

B. LANDMARK

The shortest path distance in a graph is a metric and satisfies the triangle inequality. That is, for any vertex $s, t, v \in V$,

$$|d(s, v) - d(v, t)| \leq d(s, t) \leq d(s, v) + d(v, t)$$

We call the right side of the inequality the upper bound of the distance between $s$ and $t$, which is denoted as $\bar{d}(s, t)$.

The upper bound equals the exact distance between $s$ and $t$ if the shortest path $p(s, t)$ passes through $v$. Thus, if we select a vertex $v$ and precompute the distance $d(v, u)$ from this vertex to each other vertex $u$ in the graph, we can obtain an estimated distance $\bar{d}(s, t) = d(s, v) + d(v, t)$ between any two vertices $s$ and $t$.

Furthermore, if we select a set $\{l_1, l_2, \ldots, l_k\}$ of $k$ vertices as landmarks, a potentially upper bound approximation can be computed as follows.

$$\bar{d}(s, t) = \min_{1 \leq i \leq k} d(s, l_i) + d(l_i, t)$$

(1)
Most previous studies [10], [14], [17], [18], [22] focus on how to select landmarks for accurately estimating the distance of two vertices. In this paper, we propose a landmark-based distance computation framework to improve the performance of evaluating the problem of SSSP length.

C. COMPUTATIONAL MODEL OF A DISTRIBUTED GRAPH SYSTEM

With the increasing scale of the graph, the distributed graph system has become the main application method in reality. The computational model of distributed graph systems is a vertex-centric model based on the BSP model that is composed of a sequence of supersteps. A user-defined function is executed at each superstep by each vertex in the graph in parallel. This function describes the operations that a vertex needs to perform in a superstep.

In particular, the computational model of distributed graph systems takes a graph G as input data. Each vertex v is in the active state or inactive state. Each vertex of G has a vertex ID and a modifiable user-defined value associated with it. In addition, each edge recording its target vertex ID is associated with its source vertex and a modifiable user-defined value. Let M be the set of messages, and the function sendMsg: V × V × M → ∅ sends messages M from one vertex v to another vertex v’. The function superStep obtains the number of the current supersteps. In the first superstep, some vertices are set to the active state. When a vertex receives a message not meeting the user-defined conditions from another vertex, this vertex is set to inactive. The user-defined function is called compute(G, M). When each vertex v executes the function compute(G, M), it (1) first receives messages (i.e., each m ∈ M) from the previous superstep, (2) obtains and/or updates its value or state, (3) modifies M to merge the set of new messages M’, and (4) invokes sendMsg(v, v’, M’) to send M’ to the adjacent vertices v’ along the adjacent edges of v. The function voteToHalt: V → ∅ makes vertices dormant under certain conditions. The entire computation terminates when all vertices are inactive.

III. A LANDMARK-BASED DISTANCE COMPUTATION FRAMEWORK OVER DISTRIBUTED GRAPH SYSTEMS

The main contribution of this paper is to use the landmark-based method to perform an efficient computation of an SSSP length query over distributed graph systems. Our idea is to first select some landmarks and compute their shortest path trees in the offline stage. Then, when a query of vertex v is input, we employ distributed graph systems, such as Pregel [16], Pregel+ [25], Giraph [21], and GraphX [9], to perform the distance computation between v and any other vertex based on the shortest path trees of the selected landmarks.

In the offline stage, we take advantage of our heuristic method to select a set of vertices as landmarks. The details of our landmark selection method are discussed in Section IV. Let L = {l1, l2, l3, · · · , lk} be the set of landmarks. Then, we calculate the shortest path tree from each landmark to the remaining vertices in the graph. All the shortest path trees computed at this stage are maintained and used for online query processing.

During the evaluation of a query starting from s, for any vertex v, the distance from s to v, d(s, v), is computed over distributed graph systems using the following steps:

- We initialize each vertex v with a distance value v.d and initialize it as +∞. We start the traversal by setting the source vertex s active in the first superstep and assign s.d as 0.
- In each superstep, the active vertices send their distances to s to their neighbors, and their neighbors become active. When a vertex v becomes active, if it is a landmark, we look up the shortest distances s.dv and v’.dv for each vertex v’ from SPT(v). If s.dv + v’.dv < v’.d, then we update v’.d as s.dv + v’.dv. If v is not a landmark, we only update v.d.
- The traversal terminates when all vertices are inactive, which indicates that all vertices’ distances to s are computed.

In practice, as the landmark-based distance computation algorithm runs, increasing distances from the source to other vertices are computed. Thus, the effect of the landmarks’ shortest path tree becomes limited. At the last moment of the distance computation, only a few vertices’ distances have not been computed. Then, when the traversal meets a landmark, it would be better to directly continue the traversal rather than load its shortest path tree. Therefore, we can count the number of shortest path trees of landmarks that are loaded, Countl, when we utilize the current landmark’s shortest path tree. If Countl is larger than a threshold θ, we stop loading the shortest path tree of the landmark even when the traversal meets a vertex v.

We propose an algorithm as shown in Algorithm 1 to implement the above ideas over distributed graph systems. We first associate each vertex with a tag visited and a value d, where visited records whether the vertex is visited and d records the distance from the source to the vertex (Lines 1-2 in Algorithm 1). We also initialize Countl as 0 (Line 3 in Algorithm 1). Then, we execute the user-defined function Compute(G, M) in distributed graph systems (Line 4 in Algorithm 1). In Function Compute(G, M), at the first superstep, only the source vertex is active. The tag visited and distance d of the source vertex are set to true and 0, respectively. The above information is added in message M, and sent to the neighbors of s (Lines 1-6 in Function Compute(G, M)). In addition, at each superstep, there are five steps in Compute(G, M) for each vertex v: (1) receives the set Mt of messages from the adjacent vertices, (2) updates its tag visited and distance d in vertex v, (3) merges the set Mt of new messages according to the updated values, (4) updates the distance values of all vertices if v is a landmark and Countl is smaller than a threshold θ, and (5) sends Mt to the adjacent vertices of v (Lines 7-25 in Function Compute(G, M)).
IV. LANDMARK SELECTION

In the algorithm description above, we optimize the distributed distance computation by considering the set of landmarks, and moreover, the set of landmarks is crucial for the performance of this method.

The best landmark to select is a vertex that is very central in the graph and many shortest paths pass through. In fact, selecting the best landmark is related to finding the vertex with the highest betweenness centrality \[8\]. The betweenness centrality of vertex \(v\) is defined as

\[
C_B(v) = \sum_{s \neq t \neq v \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}
\]  

(2)

where \(\sigma_{st}\) is the number of shortest paths from \(s\) to \(t\) and \(\sigma_{st}(v)\) is the number of shortest paths from \(s\) to \(t\) that \(v\) lies on.

In real graphs, although some vertices have large betweenness values, the shortest paths that they lie on highly overlap. The more shortest paths that on which two vertices lie overlap, the fewer shortest paths they can totally lie on. If two vertices lie on a similar set of shortest paths in the graph, it is probably wise to include only one of them.

For example, let us consider the two vertices \(v_6\) and \(v_{12}\) in our example graph in Figure 1. It is obvious that the shortest paths passing through \(v_6\) (or \(v_{12}\)) also pass through \(v_{12}\) (or \(v_6\)), so the shortest paths that they lie on highly overlap. Hence, although both of them may have high values of betweenness, we should only select one of them as the landmark.

To achieve the goals above, we extend the definitions in \[7\], \[15\] to define the set betweenness of \(k\) vertices by extending the definition of betweenness of a vertex to the number of shortest paths that the \(k\) vertices lie on to measure the benefit of selecting them as landmarks. Given a set of \(k\) vertices \(L = \{v_1, v_2, \ldots, v_k\}\), the set betweenness of \(L\) is defined as follows.

\[
C_B(L) = \sum_{v \in L} \frac{\sigma_{st}(v)}{\sigma_{st}}
\]  

(3)

As suggested in Equation 3, we should select the set of \(k\) vertices with the largest set betweenness as landmarks to cover as many shortest paths as possible. However, we can prove that the problem of selecting the set of \(k\) vertices with the largest set betweenness is NP-hard in the following theorem.

**Theorem 1**:  The problem of selecting the set of \(k\) vertices with the largest set betweenness as landmarks is NP-hard.

**Proof**:  Here, we first prove that the function of set betweenness, \(C_B(L)\) in Equation 3, is submodular. Here, let \(\Delta_B(v|L) = C_B([v] \cup L) - C_B(L)\) be the discrete derivative of set betweenness at \(L\) with respect to \(v\). In other words, for every \(V_1 \subseteq V_2\) and a vertex \(v \notin V_2\), we need to prove that \(\Delta_B(v|V_1) \geq \Delta_B(v|V_2)\).

For vertex \(v\), there are three kinds of shortest paths on which \(v\) lies: the set \(SP_1\) of shortest paths on which any vertices in \(V_2\) do not lie, the set \(SP_2\) of shortest paths on which only some vertices in \((V_2 - V_1)\) lie, and the set \(SP_3\) of shortest paths on which some vertices in \(V_1\) lie.

Since any shortest paths in \(SP_1\) and \(SP_3\) do not concern vertices in \((V_2 - V_1)\), the marginal gains of \(v\) for \(V_1\)
and $V_2$ over $SP_1$ and $SP_3$ are the same. However, for $SP_2$, the shortest paths in $SP_2$ passing through $v$ are not considered when computing $C_B(V_1)$ but are considered when computing $C_B([v] \cup V_1)$. In contrast, the shortest paths in $SP_2$ passing through $v$ are considered when both computing $C_B(V_2)$ and $C_B([v] \cup V_2)$. Hence, $\Delta_B([v] \cup V_1) > \Delta_B([v] \cup V_2)$.

In conclusion, $\Delta_B([v] \cup V_1) \geq \Delta_B([v] \cup V_2)$ and the function $C(L)$ is submodular. Since the problem of maximizing submodular functions is NP-hard [4], the problem is NP-hard. □

A. OUR SOLUTION

As proven in Theorem 1, selecting the set $L$ of $k$ vertices with the largest set betweenness as landmarks is an NP-complete problem. We propose a greedy algorithm as outlined in Algorithm 2.

In general, for each vertex $v$, we initialize a set $SP(v)$ that denotes the set of shortest paths on which $v$ lies (Lines 1-2 in Algorithm 2). Then, we first compute all vertices’ shortest path trees and traverse each shortest path tree to obtain $SP(v)$ for each vertex $v$ (Lines 4-9 in Algorithm 2). Note that computing all vertices’ shortest path trees can also use our proposed algorithm in distributed graph systems (Algorithm 1). During computing the shortest path tree of a sample vertex $v$, we can temporarily deem the sample vertices whose shortest paths have been computed as landmarks, and then call Algorithm 1 to compute $v$’s shortest path trees (Line 6 in Algorithm 2). Afterward, we iteratively select the vertex $v_{\text{max}}$ to maximize the marginal gain of the set betweenness of selected landmarks until we meet the constraint of the number of the landmarks or cannot find a landmark to increase the set betweenness (Lines 11-15 in Algorithm 2). Finally, the algorithm outputs $L$ (Line 16 in Algorithm 2).

**Algorithm 2: Landmark Selection**

Input: A graph $G = (V, E)$

Output: A set $L \subseteq V$ of $k$ vertices as landmarks

1. for each vertex $v$ in $V$
2.     $SP(v) \leftarrow \emptyset$;
3.     $S \leftarrow \emptyset$;
4. for each vertex $v$ in $V$
5.     Add $v$ in $S$;
6.     Use $S$ as the landmarks to call Algorithm 1 to compute the shortest path tree $SPT(v)$ rooted at $v$;
7. for each shortest path $p(v, v')$ from $v$ to $v'$
8.     for each vertex $v''$ in $p(v, v')$
9.         Insert $p(v, v'')$ in $SP(v'')$;
10. $L \leftarrow \emptyset$;
11. for $i = 1$ to $k$
12.     Find the vertex $v_{\text{max}}$ with the largest size of $SP(v)$;
13.     $L \leftarrow L \cup \{v\}$;
14. for each vertex $v$ in $V$
15.     $SP(v) \leftarrow SP(v) - SP(v_{\text{max}})$;
16. Return $L$;

Theorem 2: Algorithm 1 obtains a set of $k$ vertices as landmarks of set betweenness at least $\frac{1}{2}(1 - \frac{1}{e})$ times the value of an optimal solution.

Proof: In Algorithm 2, the problem of selecting the set of $k$ vertices with the largest set betweenness as landmarks is a problem of maximizing a submodular set function subject to a constraint as discussed in Theorem 1. We directly apply the greedy algorithm in [12] to iteratively select the vertex with the largest set betweenness. In [12], the authors prove that the worst-case performance guarantee of the greedy algorithm is $\frac{1}{2}(1 - \frac{1}{e})$, so the set betweenness of the selected landmarks is at least $\frac{1}{2}(1 - \frac{1}{e})$ of the optimal set betweenness. □

In real applications, there are often too many vertices. Then, computing the shortest path trees of all vertices is too costly. Hence, we can sample some vertices and compute their shortest path trees to estimate the set betweenness of vertices.

B. ANALYSIS

In this section, we analyze the space and time complexities of our proposed method.

We first analyze the space complexity of our method. Each landmark needs to store its distances to all vertices. We assume that the number of selected landmark is $k$, so the total number of entries to be stored is $O(k \times |V|)$, which indicates that the space complexity is $O(k \times |V|)$.

Next, we analyze the time complexity of precomputation. We can divide the landmark selection in precomputation (Algorithm 2) into two stages: the first stage is sampling a set of vertices and computing the approximate set of shortest paths on which each vertex lies; and the second stage is selecting the optimal set of landmarks.

For the first stage, we assume that the numbers of sample vertices is $s$ and the diameter (i.e. longest shortest path) of the graph is $d$. In distributed graph systems, for each sample vertex, it takes at most $d$ supersteps to compute its shortest path tree and find the shortest path between it and any other sample vertices. Each superstep involves all edges at most. Hence, the time complexity of the first stage is $O(s \times d \times |E|)$.

On the other hand, for the second stage, the time complexity of finding the vertex with largest size of $SP(v)$ is $O(|V|)$; and the time complexity of recomputing the set of shortest paths on which each remaining vertex lies is still $O(V)$. We should select $k$ landmarks, so the time complexity of the second stage is $O(k \times |V|)$. In summary, the time complexity of precomputation is $O(s \times d \times |E| + k \times |V|)$.

Finally, we analyze the time complexity for processing queries. In the worst case, no landmark is used to speed up the computation and we do a full traversal over the whole graph, so it takes at most $d$ supersteps and each superstep involves all edges at most. Hence, the time complexity of processing queries is $O(d \times |E|)$.

V. EXPERIMENTS

In this section, we evaluate the performance of our method on several real graphs and on different distributed graph systems.
A. SETUP
We test our method on the following datasets that exist in the real world and can be classified in knowledge graphs, road networks and social networks.

1) DBpedia
DBpedia [3] is a crowd-sourced RDF knowledge graph derived from Wikipedia. It can also be represented by a graph, where resources are vertices and relationships between resources are edges.

2) YAGO
YAGO [19] is an RDF knowledge graph that mainly integrates data from Wikipedia, WordNet, and GeoNames. It can be represented as a graph, where entities are vertices and relationships between entities are edges.

3) SOC-ACADEMIA
Soc-Academia [20] is a social network extracted from Academia.edu, a platform for academics to share research papers. In this network, each vertex corresponds to a researcher. An undirected edge between two authors represents a link between them.

4) Soc-YouTube
Soc-YouTube [20] is a social network extracted from YouTube, a video-sharing website that includes a social network. In this network, each vertex corresponds to a user. An undirected edge between two authors represents a link between them.

5) RoadNet-TX
RoadNet-TX [13] is a dataset describing the road network of Texas. Intersections and endpoints in Texas are represented by vertices and the roads connecting these intersections or endpoints are represented by edges.

6) RoadNet-CA
RoadNet-CA [13] is a dataset describing the road network of California. Similarly, intersections and endpoints are vertices and the roads are edges.

These datasets and their properties are listed in Table 1, where |V|, |E| and Δ are the numbers of vertices, edges and triangles, respectively, δ represents the average clustering coefficient and d expresses the diameter (i.e. the longest shortest path).

All experiments are conducted on a cluster with 12 physical nodes in the Alibaba Cloud. Each node has four CPUs with 32 GB memory and a 100 GB disk drive. We test our proposed techniques over multiple distributed graph systems, including Pregel+ [25], Giraph [21] and GraphX [9]. Pregel+ has the best scalability and Giraph [21] and GraphX [9] fail over road networks, so we use Pregel+ in different experiments by default.

By default, we sample 500 vertices to estimate the value of the set betweenness of each vertex and select 100 landmarks from them. When we evaluate the SSSP length query performance, we set the threshold θ of Countl discussed in Section III as 30. Even for one query, our method can speed up its performance and the response times of different queries spread out over a small range. Thus, we sample 500 vertices from each graph as the input distance queries to evaluate different methods and evaluate them one by one. Then, we report the average response time per query.

B. COMPARISON WITH DIFFERENT LANDMARK SELECTION METHODS
We compare our method defined as set betweenness with some classic landmark selection methods. Other methods involved in the assessment include random (denoted as RN), degree (denoted as Degree) [5], betweenness (denoted as BN) [8], and coverage (denoted as Coverage) [14]. We also selected 100 landmarks from them. Random is a landmark selection strategy. It randomly selects vertices as landmarks in the graph to perform the calculation process. Degree is a method of selecting the vertex with the largest degree in the graph as the landmark. Betweenness is a strategy of selecting a series of vertices with the highest probability of appearing in the shortest path between pairs of vertices as landmarks. The idea of coverage is to iteratively choose vertices as landmarks having the approximate largest coverage, which represents the number of vertex pairs that the shortest paths from the vertex to other vertices pass through in the graph. The landmark selection based on the measure of set betweenness is denoted as SB. We also compare with a baseline where a distance query is not optimized based on landmarks but still computed on the distributed graph system (denoted as Baseline).

Figure 3 summarizes the average query response time of a query over various landmark selection methods on all six datasets. We find that the measure of set betweenness has superior performance in the optimization of distance computational time compared with other measures, especially for road networks. The diameters of road networks are much larger than those of the other two kinds of graphs, which indicates that it takes many more iterations for distance computation in road networks than other kinds of graphs. Our landmark-based distance computation frameworks can greatly reduce the number of iterations to improve performance, especially when there are many iterations for road networks.

C. EFFECT OF LANDMARKS’ NUMBER
In this experiment, we study the performance of the method in this paper when different numbers of landmarks are selected. As shown in Figure 4, we summarize the optimized distance computational time for six datasets varying the number of landmarks from 100 to 500. We can intuitively find that with the increase in the number of landmarks, the average query response time has a tendency to decrease. This is because when computing the distance online, some landmarks can...
be selected to participate in the computation. The larger the number of landmarks is, the larger the range of landmarks leading to the nearest landmarks, in theory, becoming closer to reality.

### TABLE 1. Datasets.

| Dataset          | $|V|$   | $|E|$     | $\delta$ | $\Delta$ | $d$  |
|------------------|-------|----------|----------|----------|------|
| Knowledge Graphs | DBpedia | 911,669  | 5,228,914 | 0.1032   | 257,029 | 12   |
|                  | YAGO   | 4,836,651| 10,318,743| 0.0851   | 283,518 | 70   |
| Social Networks  | Soc-Academia | 200,169 | 2,045,324 | 0.241358 | 8,407,098 | 11  |
|                  | Soc-YouTube | 495,957 | 1,936,748 | 0.11008  | 7,343,398 | 14  |
| Road Networks    | RoadNet-TX | 1,379,917 | 1,921,660 | 0.0470   | 82,869  | 1,054 |
|                  | RoadNet-CA | 1,965,206 | 2,766,607 | 0.0464   | 120,676 | 849  |

### FIGURE 3. Comparison of different landmark selection methods.

### FIGURE 4. Evaluating different number of landmarks.

**D. EFFECT OF $\theta$**

As discussed in Section III, as the landmark-based distance computation algorithm runs, the effect of the landmarks' shortest path tree becomes limited and we define a threshold
We stop loading the shortest path tree of the landmark during distance computation if the number of landmarks’ shortest path trees that have been loaded is more than \( \theta \). In this experiment, we evaluate the effect of \( \theta \). Figure 5 shows the experimental results. As \( \theta \) gradually increases from 10 to 90, we find that the trend of time after optimization basically decreases first and then increases, and there is an inflection point. This indicates that the effect of landmarks’ shortest path trees used in distance computation is initially large and then decreases after an inflection point.

E. PERFORMANCE OVER DIFFERENT DISTRIBUTED GRAPH SYSTEMS

In this experiment, we conduct experiments to evaluate the performance of our proposed techniques on different distributed graph systems, including Pregel+ [25], Giraph [21] and GraphX [9]. Note that because the diameters of road networks are so large that Giraph [21] and GraphX [9] fail on them, we only compare these distributed graph systems on knowledge graphs and social networks. As shown in Figure 6, Pregel+ has better overall performance than Giraph and GraphX. Pregel+ uses less memory and can be 2-3 times faster than a Java implementation.

F. OFFLINE PERFORMANCE

In this experiment, we conduct an experimental evaluation of the precomputation time of the proposed method in this paper. We divide our landmark selection into two stages: sampling a set of vertices and computing the approximate set of shortest paths on which each vertex lies and selecting the optimal set of landmarks. We can clearly discover from Table 2 that for small social networks such as Soc-Academia, the landmarks can be obtained within approximately 666 minutes, while the largest road networks (RoadNet-CA) require approximately 6, 133 minutes of preprocessing.

VI. RELATED WORK

A. LANDMARK-BASED DISTANCE COMPUTATION

Landmark-based distance computation has been widely studied in previous studies [1], [2], [10], [14], [17], [18], [22], [27]. Potamias et al. [17] first evaluated the landmark-based algorithm for approximate distance estimation in large graphs. The algorithms in [10] extended the above method by storing complete shortest paths to each landmark at each vertex. Qiao et al. identified a local landmark close to the specific query vertex [18]. Tretyakov [22] discussed how to maintain the information of landmarks incrementally under edge insertions and deletions. Valstar et al. [23] discussed how to utilize landmarks to optimize regular simple path query processing on large graphs. Li et al. [14] proposed a new measure named coverage to optimize the lower bound of landmark-based distance estimation. Akiba et al. [1], [27], Yano et al. [27] proposed a pruned landmark labeling framework to evaluate the exact distance queries, approximate distance queries and distance queries over time-evolving graphs.

Betweenness centrality and its variants have also been studied and many papers discuss how to compute them in a single machine [7], [15]. Fink and Spoerhase [7] first give a reduction of the problem of computing betweenness to the problem of budgeted maximum coverage, and then design a new algorithm with an approximation factor based on the reduction. Mahmoody et al. [15] present a randomized algorithm based on sampling shortest paths while providing theoretical guarantees. In this paper, we use the concept of set betweenness and design a heuristic distributed algorithm to compute the set betweenness of a large graph.
TABLE 2. Offline performance (in min).

|                  | Sampling and Computing | Selecting Landmark | Total Time |
|------------------|------------------------|--------------------|-----------|
| DBpedia          | 265                    | 620                | 885       |
| YAGO             | 651                    | 1,822              | 2,473     |
| Soc-Academia      | 187                    | 479                | 666       |
| Soc-YouTube       | 233                    | 779                | 1,012     |
| RoadNet-TX        | 2,287                  | 3,039              | 5,326     |
| RoadNet-CA        | 2,644                  | 3,489              | 6,133     |

B. DISTRIBUTED GRAPH PROCESSING SYSTEMS

As an increasing number of real applications concern large graphs, many distributed graph systems have been proposed. The computation model of these systems is a vertex-centric model that is composed of a sequence of supersteps conforming to the bulk synchronous parallel (BSP) model. The first distributed graph processing system was Pregel [16].

After Pregel, many systems were proposed to optimize it. Giraph [21] is the open-source counterpart to Pregel; GraphX [9] is Spark’s graphs computing API, which also implements Giraph [21] is the open-source counterpart to Pregel; GraphX [9] is Spark’s graphs computing API, which also implements Pregel. Pregel++ [11] discusses how to process large graphs by exploiting external memory without compromising efficiency; GraphD [26] adopts a semistreaming model to avoid scanning the whole graph in each superstep, where only a portion of the vertex states are maintained in the main memory and edges and messages are streamed on the local disk; G-thinker [24] extends the vertex-centric model to the subgraph-centric model for compute-intensive graph mining workloads.

VII. CONCLUSION

In this paper, we propose a distributed landmark-based distance computation framework for using distributed graph systems to compute the distances from a source to all vertices in large graphs. We use a measure set betweenness to select a sequence of approximate optimal landmarks, which plays an important role in speeding up distance calculations. Although we prove that finding the optimal set of landmarks is NP-complete, we propose a heuristic distributed algorithm with a bounded approximation ratio. Finally, we verify that our algorithm is effective and efficient for computing the distance between two vertices in large graphs by extensive experiments.

REFERENCES

[1] T. Akiba, Y. Iwata, and Y. Yoshida, “Fast exact shortest-path distance queries on large networks by pruned landmark labeling,” in Proc. Int. Conf. Manage. Data (SIGMOD), 2013, pp. 349–360.
[2] T. Akiba, Y. Iwata, and Y. Yoshida, “Dynamic and historical shortest-path distance queries on large evolving networks by pruned landmark labeling,” in Proc. 23rd Int. Conf. World Wide Web (WWW), 2014, pp. 237–248.
[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, “DBpedia: A nucleus for a Web of open data,” in The Semantic Web, Berlin, Germany: Springer, 2007, pp. 722–735.
[4] L. Bordeaux, Y. Hamadi, and P. Kohli, “Tractability: Practical Approaches to Hard Problems.” Cambridge, U.K.: Cambridge Univ. Press, 2014.
[5] S. P. Borgatti, “Centrality and network flow,” Social Netw., vol. 27, no. 1, pp. 55–71, Jan. 2005.
[6] J. Cheng, Y. Ke, S. Chu, and C. Cheng, “Efficient processing of distance queries in large graphs: A vertex cover approach,” in Proc. Int. Conf. Manage. Data (SIGMOD), 2012, pp. 457–468.
[7] F. Mink and J. Spoerhase, “Maximum betweenness centrality: Approximability and tractable cases,” in Proc. WACOM, 2011, pp. 9–20.
[8] L. C. Freeman, “A set of measures of centrality based on betweenness,” Sociometry, vol. 40, no. 1, pp. 35–41, Mar. 1977.
[9] J. E. Gonzalez, R. S. Xin, A. Dave, D. Crankshaw, M. J. Franklin, and I. Stoica, “GraphX: Graph processing in a distributed dataflow framework,” in Proc. OSDI, 2014, pp. 599–613.
[10] A. Gubichev, S. Bedathur, S. Seufert, and G. Weikum, “Fast and accurate estimation of shortest paths in large graphs,” in Proc. 19th ACM Int. Conf. Inf. Knowl. Manage. (CIKM), 2010, pp. 499–508.
[11] S. Ko and W. Han, “TurboGraph++: A scalable and fast graph analytics system,” in Proc. SIGMOD, 2018, pp. 395–410.
[12] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. Van Briesen, and N. Glance, “Cost-effective outbreak detection in networks,” in Proc. KDD. New York, NY, USA: Association for Computing Machinery, 2007, pp. 420–429.
[13] J. Leskovec and A. Krevl. (Jun. 2014). SNAP Datasets: Stanford Large Network Dataset Collection. [Online]. Available: http://snap.stanford.edu/data
[14] M. Li, P. Peng, Y. Xu, H. Xia, and Z. Qin, “Distributed landmark selection for lower bound estimation of distances in large graphs,” in Proc. APWeb-WAIM, 2019, pp. 223–239.
[15] A. Mahmood, C. E. Tsourakakis, and E. Upfal, “Scalable betweenness centrality maximization via sampling,” in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2016, pp. 1765–1773.
[16] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, “Pregel: A system for large-scale graph processing,” in Proc. Int. Conf. Manage. Data (SIGMOD), 2010, pp. 135–146.
[17] M. Potamias, F. Bonchi, C. Castillo, and A. Gionis, “Fast shortest path distance estimation in large networks,” in Proc. 18th ACM Conf. Int. Conf. Knowl. Manage. (CIKM), 2009, pp. 867–876.
[18] M. Qiao, H. Cheng, L. Chang, and J. X. Yu, “Approximate shortest distance computing: A query-dependent landmark scheme,” IEEE Trans. Knowl. Data Eng., vol. 26, no. 1, pp. 55–68, Jan. 2014.
[19] T. Rebete, F. M. Suchanek, J. Hoffart, J. Biega, E. Kuzey, and G. Weikum, “YAGO: A multilingual knowledge base from Wikipedia, Wordnet, and Geonames,” in Proc. ISWC, 2016, pp. 177–185.
[20] R. A. Rossi and N. K. Ahmed, “The network data repository with interactive graph analytics and visualization,” in Proc. AAAI, 2015.
[21] S. Sakr, F. M. Orakzai, I. Abdelaziz, and Z. Khayyat, Large-Scale Graph Processing Using Apache Giraph. Cham, Switzerland: Springer, 2016.
[22] K. Tretjakov, A. Armas-Cervantes, L. García-Baquales, J. Vilo, and M. Dumas, “Fast fully dynamic landmark-based estimation of shortest path distances in very large graphs,” in Proc. 20th ACM Int. Conf. Inf. Knowl. Manage. (CIKM), 2011, pp. 1785–1794.
[23] L. D. J. Valstar, G. H. L. Fletcher, and Y. Yoshida, “Landmark indexing for evaluation of label-constrained reachability queries,” in Proc. ACM Int. Conf. Manage. Data (SIGMOD), 2017, pp. 345–358.
[24] D. Yan, H. Chen, J. Cheng, M. T. Özsu, Q. Zhang, and J. C. S. Lui, “G-thinker: Big graph mining made easier and faster,” in Proc. ICDE, 2020.
[25] D. Yan, J. Cheng, Y. Lu, and W. Ng, “Effective techniques for message reduction and load balancing in distributed graph computation,” in Proc. 24th Int. Conf. World Wide Web (WWW), 2015, pp. 1307–1317.
Q. Wang et al.: Optimizing Distance Computation in Distributed Graph Systems

[26] D. Yan, Y. Huang, M. Liu, H. Chen, J. Cheng, H. Wu, and C. Zhang, “Graphd: Distributed vertex-centric graph processing beyond the memory limit,” IEEE Trans. Parallel Distrib. Syst., vol. 29, no. 1, pp. 99–114, Aug. 2018.

[27] Y. Yano, T. Akiba, Y. Iwata, and Y. Yoshida, “Fast and scalable reachability queries on graphs by pruned labeling with landmarks and paths,” in Proc. 22nd ACM Int. Conf. Conf. Inf. Knowl. Manage. (CIKM), 2013, pp. 1601–1606.

[28] Q. Zhang, D. Yan, and J. Cheng, “Quegel: A general-purpose system for querying big graphs,” in Proc. Int. Conf. Manage. Data (SIGMOD), 2016, pp. 2189–2192.

QING WANG is currently pursuing the Ph.D. degree with the College of Computer Science and Electronic Engineering, Hunan University, China. His main research interests include distributed graph management and NoSQL database.

SHENGYI JI is currently pursuing the B.E. degree with the Department of Software Engineering, Hunan University. He is also a Research Assistant under the supervision of Dr. P. Peng. His research interests include graph database and distributed RDF systems.

PENG PENG (Member, IEEE) received the B.S. degree from Beijing Normal University, in 2009, and the Ph.D. degree in computer science from Peking University, in 2016. He is currently an Associate Professor with Hunan University. His research interests include graph database and distributed RDF systems.

MINGDAO LI received the B.S. degree from Hunan University, in 2017, where he is currently pursuing the Ph.D. degree in computer science and technology under the supervision of Dr. P. Peng and Dr. Z. Qin. His research interests include graph database and community search over big graphs.

PING HUANG is currently pursuing the M.S. degree with the Department of Software Engineering, Hunan University, under the supervision of P. Peng. Her research interests include graph database, graph computation, and distributed RDF systems.

ZHENG QIN (Associate Member, IEEE) received the Ph.D. degree in computer software and theory from Chongqing University, China, in 2001. He has rich experience in products development and application services, such as financial, medical, military, and education sectors. He is currently a Professor of computer science and technology with Hunan University, China. His main research interests include computer networks and information security, cloud computing, big data processing, and software engineering. He is a member of the China Computer Federation (CCF) and ACM.