GeoPregel: An End-to-End System for Privacy-Preserving Graph Processing in Geo-Distributed Data Centers

Abstract—**Graph processing is a popular computing model for big data analytics. Emerging big data applications are often maintained in multiple geographically distributed (geo-distributed) data centers (DCs) to provide low-latency services to global users. High performance graph processing in geo-distributed DCs is often hard to achieve due to scarce and costly inter-DC network resources. Furthermore, due to increasing privacy concerns, geo-distribution imposes diverse, strict, and often asymmetric, privacy regulations that constrain geo-distributed graph processing. Existing graph processing systems fail to address these two challenges. In this paper, we design and implement *GeoPregel*, which is an end-to-end system that provides *privacy-preserving* graph processing in geo-distributed DCs with *low latency* and *high utility*. To ensure privacy, GeoPregel smartly integrates Differential Privacy into graph processing systems with the help of two core techniques, namely *sampling* and *combiners*, to reduce the amount of inter-DC data transfer while preserving good accuracy of graph processing results. We implement our design in Giraph and evaluate it in real cloud DCs. Results show that GeoPregel can preserve the privacy of graph data in geo-distributed DCs with high performance and good accuracy.**

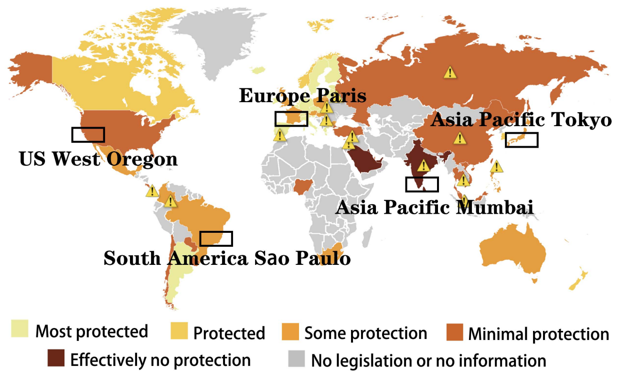


Fig. 1 The privacy and data protection restrictions of 54 countries (Forrester's data privacy heatmap released in 2021 [8]).

I. Introduction

Graph processing is a popular computing model to perform big data analytics for a wide range of applications. With the ever increasing sizes of data, many big data applications need to analyze large-scale graph data generated and maintained globally. For example, the social network graph at Facebook has over one trillion edges [1], [2]. In order to provide reliable and low-latency services to the users, Facebook has built multiple geographically distributed (geo-distributed) data centers (DCs) to maintain and manage those data. To run graph processing algorithms on top of such data, for example to study the importance of Facebook users using recommendation algorithms, coordination (e.g., data exchange) is needed among multiple geo-distributed DCs. Geo-distributed data communication leads to two challenges which make existing graph processing engines inefficient or even invalid.

First, achieving good performance for geo-distributed graph processing is challenging since wide area network (WAN) resources are scarce and costly. Our measurements on real cloud DCs show that inter-DC network bandwidths can be lower than one tenth of the intra-DC network bandwidths (see Section II). On the other hand, geo-distributed graph processing often leads to large amount of inter-DC data communications. For example, when running the PageRank algorithm on Twitter graph using the real geographical distribution of Twitter users [3], over 70% of the total communications are inter-DC ones. Thus, it is important to reduce inter-DC data transfer size to achieve good performance for geo-distributed graph data analytics. Existing studies tried to achieve this goal through optimized graph partitioning and subgraph placement [4], [5]. However, this may require costly data movement between DCs and can hardly adapt to dynamic graph changes. A few studies have looked at the problem from system perspective and proposed new graph computation model to avoid inter-DC data transfer [6], [7]. However, such systems may sacrifice computation accuracy or convergence speed for good performance and are difficult to generalize for different applications [6].

Second, apart from the performance issue, data privacy is another important concern that makes existing graph processing systems practically and legally not useful. Many countries and regions have strict laws and regulations to protect the privacy of personal data. For example, the states of the European Union (EU) enforces the General Data Protection Regulations (GDPR) [9] to protect personal data of individuals living in the EU. According to GDPR, which is considered as the most comprehensive privacy protection regulation to date, it is *permitted and legal* to transfer personal data out of EU only when the target country has "adequate" level of data protection [10]. However, different countries can have different views on the importance of data privacy. Figure 1 shows the different levels of data privacy protection in 54 countries around the world. As a result, it is illegal for international companies such as Facebook to transfer its user data collected in the EU to the US DCs [11] without protection, which casts new challenge to geo-distributed data analytics.

Most existing graph processing systems for geo-distributed DCs have overlooked the privacy issue [6], [7]. Although there have been some privacy-preserving systems for applications such as stream processing [12] and machine learning [13], these systems cannot be easily adapted to graph applications which have different computation and communication patterns. On the other hand, although privacy-preserving graph publishing can be used to protect the privacy of individuals [14], [15], [16], [17], [18], there still lacks an end-to-end solution which can preserve graph data privacy during geo-distributed execution without modifying the applications.

Motivated by the above, our goal in this paper is to design *GeoPregel*, a *privacy-preserving* graph processing system for geo-distributed DCs. Although a number of techniques such as data encryption [19], [20], [21] and secure multi-party computation [22], [23] can be used to preserve data privacy, they usually have high latency and can harm the performance of graph systems. *Differential Privacy (DP)* [24] protects the privacy of individuals by adding random noise to aggregated data. Satisfying it by perturbing the outgoing inter-DC messages would allow DCs to perform all computations over cleartext (perturbed) data, thus resulting in lightweight graph applications without jeopardizing privacy guarantees. However, naively integrating DP into existing graph processing systems would add too much noise and harm the utility of graph processing results. Thus, the main technical problem addressed by this work is *how to practically integrate DP into graph processing systems for* ***good utility*** *and* ***low latency***.

We address this problem using two techniques. First, we employ *sampling* at the vertex level to reduce the number of outgoing messages and thus reducing inter-DC data transfer and hence system latency. Sampling additionally benefits utility through its well-known amplification effect for DP [25], [26], [27], [28], [29], [30], [31]. We rigorously prove that our sampling technique can amplify the privacy budget and as a result improve the utility of our graph processing system. Second, we use *combiners* to combine outgoing messages at the DC level to further reduce inter-DC data transfer. Combiners also benefit DP in two aspects. For one thing, with a smaller number of combined messages, we can assign a larger privacy budget to each message and hence improve the accuracy of graph processing results. For another, aggregated messages (e.g., summed up) reduce the impact of the differentially private perturbation, further increasing the accuracy of results. We identify and make use of the multi-level and asymmetric feature of privacy constraints in geo-distributed DCs to perturb only the messages sent to DCs with lower privacy levels and consequently further improve system utility.

We focus on recommendation algorithms in this paper, which are an important type of graph applications usually involve global data distributions [32], [33]. We evaluate the effectiveness of GeoPregel using four recommendation algorithms on real graphs and real user distributions [3]. Results show that GeoPregel can preserve the privacy of graph data in geo-distributed DCs with high performance and good accuracy.

To summarize, this paper makes the following contributions.

- We present an end-to-end system named *GeoPregel* specifically tailored for differentially private geo-distributed graph processing. *To the best of our knowledge, this is the first study that considers system latency, utility and data privacy at the same time for geo-distributed graph processing.*
- We present sampling and combiners techniques to make DP *practical* in our system. The two techniques can benefit both of our two goals: mitigating the impact of differentially private perturbation on system utility and improving system performance by reducing inter-DC data communications.
- Our extensive experiments clearly demonstrate the effectiveness and efficiency of GeoPregel for large-scale geo-distributed graph algorithms, e.g., recommendation systems. We have integrated our design into Giraph and open sourced it at: https://github.com/GeoPregel/GeoPregel.

II. Background and Motivation

A. Geo-Distributed Data Centers

We consider several DCs owned by a single entity (e.g., a social network provider) and distributed globally over distinct geographical areas possibly ruled by distinct privacy laws. We consider as a geographical area the areas that share the same legal right-to-privacy framework. Privacy laws can constrain many aspects regarding personal data, such as the collection, storage and processing of the data. In this work, we mainly focus on the aspect regarding data transfer across DCs [10]. The privacy laws allow data transfers from a DC with lower (i.e., less strict) privacy protection level to DCs with higher protection levels, but not the other way around.

The entity that owns the multiple DCs can perform global data analytical jobs that require coordination between the DCs. Data communication between geo-distributed DCs usually goes through the WAN, which has low bandwidth and high latency compared to network performance within a single DC. For example, our measurements on both Amazon EC2 and Windows Azure clouds show that the network bandwidth within a single DC can be 7-22x higher than that between geo-distributed DCs.

B. Graph Data Analytics

A number of graph processing systems such as Pregel [34] and PowerGraph [35] have been proposed to efficiently perform graph data analytics. Most of these systems follow the "think-as-a-vertex" philosophy and encode graph computation as vertex programs which run in parallel and communicate through the edges of graphs. Vertices iteratively update their states according to the messages received from neighbors. Thus, efficient communication between vertices and their neighbors is important to performance of graph processing. In this paper, we focus on the Pregel [34] model, which has a clear abstraction based on *message passing* and thus can ease our discussion on data communication optimizations.

With the global distribution of graph data sets, inter-DC data communications play an important role in geo-distributed graph processing. We study the distributions of high degree vertices ($> 10,000$ followers) in the Twitter graph using real

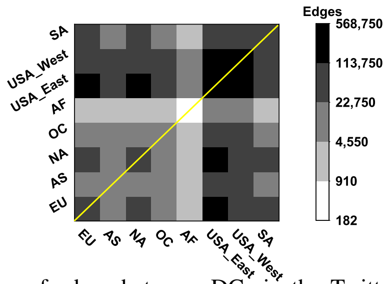


Fig. 2 Number of edges between DCs in the Twitter graph, where vertices are located in eight different DCs.

geographic locations of users [36]. Specifically, we cluster user locations into eight DCs, including South America, USA West, USA East, Africa, Oceania, North America, Asia and Europe. Figure 2 shows the number of edges between different pairs of DCs, where the diagonal cells represent intra-DC edges and the rest represent inter-DC edges. The number of edges can reflect the amount of data transfer between DCs. We find that over 75% of all edges are inter-DC edges. This leads to two technical challenges to geo-distributed graph processing.

Challenge 1: Preserving privacy with good utility. As mentioned above, geo-distributed DCs usually have different rules and regulations on data privacy protection. This casts special constraints to inter-DC data transfer. Violations of data privacy laws and regulations can lead to huge fines [11]. Thus, it is important for future graph processing systems to preserve data privacy in geo-distributed DCs. One straightforward way to achieve this goal is to forbid any inter-DC data communication that violates privacy requirements. However, this can greatly harm the accuracy of graph processing results due to the large amount of inter-DC data communications. Another natural approach could be to encrypt messages before they leave their DCs. For example, (some) fully homomorphic encryption schemes [37], [38], [39] support (limited or no) computations over encrypted data. However, the resulting overhead in computing time would be prohibitive.

In this paper, we adopt differential privacy (DP) which allows computations over cleartext data while still satisfying sound privacy guarantees. A straightforward way of applying DP is to perform differentially private perturbation on the messages exchanged during graph processing. However, this leads to unacceptable accuracy loss because of the numerous inter-DC data communications. The challenge thus translates into making graph applications satisfy differential privacy without thwarting the quality of their results.

Challenge 2: Improving graph processing performance. Due to the limited inter-DC network bandwidth, the large amount of inter-DC data communications in geo-distributed graph processing have become the major performance bottleneck. To optimize system performance, one straightforward idea is to avoid inter-DC data communications as much as possible, which however can impact the accuracy of graph processing results and thus contradicts the utility optimization goal. Some studies perform message rerouting to make usage of high bandwidth links in geo-distributed DCs to improve system performance [5]. However, this may not work when considering privacy constraints. Due to the complicated and irregular data communication patterns in graph processing, it is non-trivial to achieve graph processing performance considering both privacy and utility.

III. Problem Overview

In this work, we consider the problem of executing targeted applications over a large graph partitioned on multiple DCs located in distinct geographical areas. The goal is to satisfy data privacy requirements while keeping low latency and high utility. In the following, we formally define the problem.

A. Data Model

Consider processing a dataset partitioned across multiple geo-distributed DCs and the full dataset is a single directed and unweighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of vertices and $\mathcal{E}$ is a set of edges. The application partitions the graph into k *graph partitions* $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$, where $\mathcal{V}_i \subset \mathcal{V}$, $\mathcal{E}_i \subset \mathcal{E}$, $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset$ and $\mathcal{E}_i \cap \mathcal{E}_j = \emptyset$ if $i \neq j$. Inter-DC edges connect two nodes belonging to different partitions. We consider in that case that a (directed) inter-DC edge belongs to the set of edges of the partition of the starting node: $(u, v) \in \mathcal{E}_i$ if $u \in \mathcal{V}_i$. For simplicity and without loss of generality, we assume in the following that there is a single DC per geographical area and we consider that each DC holds a single partition of $\mathcal{G}$.

B. Targeted Graph Applications

In this work, we target four graph algorithms that are widely adopted in recommendation systems, including PageRank [40], HITS [41], SALSA [42] and ALS [43].

PageRank was originally proposed by Google to measure the importance of different webpages according to the link relationship between the pages. At each graph processing iteration, each vertex updates its importance (i.e., rank value) using the rank values of its neighbors from the last iteration.

HITS is another algorithm used to rank webpages relevant for a topic search. Given a topic search, the set of highly relevant webpages are called *Authorities* and webpages that are not very relevant but point to many related authorities are called *Hubs*. The algorithm strives to obtain authority and hub scores for each vertex. In each iteration, the two scores are updated in a mutually recursive manner. We normalize the scores to $[1, 2]$ instead of $[0, 1]$ as in the original algorithm [41] to achieve faster convergence while keeping the same ranking.

SALSA has been shown to be one of the most effective link analysis algorithms [44] and has been applied in social network recommendations [45]. Similar to HITS, SALSA divides webpages into hubs and authorities and constructs a bipartite graph by putting hubs on one side and authorities on the other. The authority/hub score of each vertex in SALSA is determined by the authority/hub scores of other vertices.

ALS is a popular matrix factorization algorithm that tries to make item recommendations to users given sparse user-item ratings. Essentially, ALS decomposes a rating matrix into the product of two lower dimensional matrices to capture the potential factors of users and items. The two matrices are alternatively updated in multiple iterations until convergence.

For simplicity, we overload the terminology when the semantics is clear from the context: PageRank may refer to the complete PageRank algorithm or to the function applied at each vertex, and the same holds for HITS, SALSA and ALS.

C. Quality Measures

Low latency and high utility are our two main goals and we measure the optimization quality of the two goals as follows.

Latency measure. As inter-DC data communication is the main performance bottleneck in geo-distributed graph processing, we use the WAN usage consumed during graph processing to measure system latency. To obtain this value, we simply aggregate the size of every message transmitted between different DCs. The size of a message is determined by the data contained in the message and can be estimated using TCP data packets.

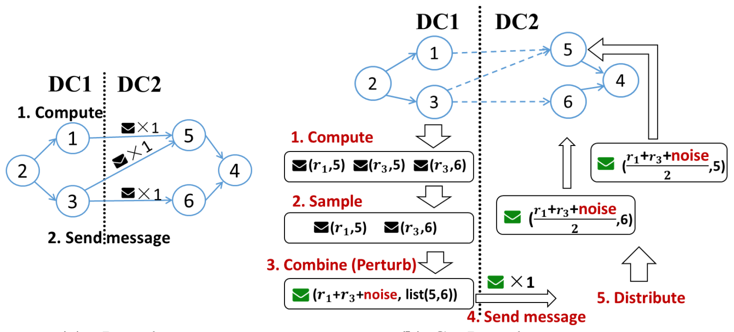
Utility measure. We quantify the impact of our techniques on the utility of the targeted graph applications based on a general error-quantification measure, i.e., the average relative error (denoted ARE), and set-based measures specifically related to recommendation systems, i.e., the precision and recall of top-k values retrieved. Note that for a top-k list, the number of false positives is equal to the number of false negatives, and consequently the precision and the recall are equal. As a result we only measure the precision below.

We measure ARE by comparing the graph processing results of our system with those from Pregel (denoted as baseline). That is, $ARE = \frac{\sum_{v \in \mathcal{V}} |(m_v - b_v)/b_v|}{|\mathcal{V}|}$, where b_v is the baseline value of vertex v and m_v is the result of vertex v obtained using our system. For precision, we sort the baseline results and the results of our system separately, and compute the true positives (denoted TP) and the false positives (denoted FP). Precision is then defined as $P = \frac{TP}{TP+FP}$.

D. Threat Model

Despite belonging to a single organization, the DCs are physically located in distinct geographical areas. This results in various and possibly asymmetric privacy requirements across DCs (see Figure 1). We capture this diversity using discretized privacy levels. When the privacy level of a DC$_i$ is lower than that of DC$_j$, it represents the real-life situation where the geographical area of DC$_j$ has privacy requirements compatible with those required by the area of DC$_i$, and it is possible to transfer raw data from DC$_i$ to DC$_j$. Note that the reverse is not necessarily true.

Our system aims to protect the privacy of information moving across DCs, i.e., the vertices in $\mathcal{V}$ that are connected to vertices located in a different DC and their corresponding edges in $\mathcal{E}$. Local information that are not shared with any other DC (i.e., the vertices that are **not** connected to any vertex located in another DC, and their corresponding edges) are not considered in our privacy protection. This protection represents the real-life situation that the graph structural information is known to the organization, while the values of vertices should be protected. For example, in user-movie recommendations,



(a) Pregel (b) GeoPregel

Fig. 3 System overview. A message contains the content and the list of receivers. For example, (r_1,5) means the message is sent to vertex 5 with a value of r_1.

the input is a bipartite graph connecting users with the movies they have rated and our model protects users' ratings to specific movies.

E. Edge Differential Privacy

To preserve data privacy, we adopt the well-known ϵ-edge-DP model [17] which essentially aims at hiding the presence/absence of any single edge. We consider that two graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ are *neighbors* if $\mathcal{G}$ is $\mathcal{G}'$ with one edge difference, i.e., $\mathcal{V} = \mathcal{V}'$ and $|\mathcal{E} - \mathcal{E}'| = 1$.

Definition 1. *[ϵ-edge-DP [17]] Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two neighboring graphs. Let* $\mathtt{f}$ *be a randomized function and* $\mathtt{Im(f)}$ *be its image (all possible outputs of* $\mathtt{f}$*). If for all $\mathcal{S} \subset \mathtt{Im(f)}$,*

$$Pr[\mathtt{f}(\mathcal{G}_1) \in \mathcal{S}] \leq e^{\epsilon} Pr[\mathtt{f}(\mathcal{G}_2) \in \mathcal{S}]$$

then $\mathtt{f}$ *is said to satisfy ϵ-edge-DP. ϵ is called privacy budget. The smaller the budget, the better the privacy.*

In this paper, $\mathtt{f} \in \{PageRank, HITS, SALSA, ALS\}$. In order to satisfy ϵ-edge-DP, it is possible to perturb their output based on the Laplace mechanism [46] parameterized with ϵ and their respective *sensitivities*. The sensitivity quantifies the maximum impact of the presence/absence of any single edge on the result of the function (detailed computation of the respective global sensitivity of the targeted applications will be provided in an appendix). Denote $\mathcal{G}_1$ and $\mathcal{G}_2$ as two neighboring graph partitions. Given a privacy budget ϵ_i and the global sensitivity $\max \|\mathtt{f}(\mathcal{G}_1) - \mathtt{f}(\mathcal{G}_2)\|_1$, satisfying ϵ_i-edge-DP requires perturbing the output of the function by adding random noise sampled from the Laplace distribution $\mathtt{Lap}(\max \|\mathtt{f}(\mathcal{G}_1) - \mathtt{f}(\mathcal{G}_2)\|_1 / \epsilon_i)$ to the results. Finally, the self-composition properties of ϵ-DP [47] result in a consumption of the privacy budget that is 1) linearly increasing with the number of perturbed outputs when the function takes as inputs overlapping graphs (i.e., sequential composition) or 2) set to the maximum privacy budget used when the function takes as inputs disjoint graphs (i.e., parallel composition).

IV. Design Details of GeoPregel

GeoPregel is designed to achieve differentially private graph processing in geo-distributed DCs with low latency and high utility. To achieve this goal, we propose two optimizations including *sampling* and *combiners* to improve the accuracy of

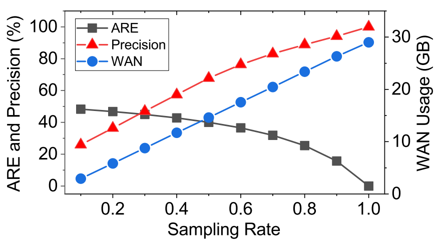


Fig. 4 Impact of sampling on the quality of graph processing results using PageRank on LiveJournal graph.

differentially-private graph computations while reducing the inter-DC data communication size. Figure 3 gives an overview of our GeoPregel model. Although most of our discussions focus on Pregel, the design ideas are general and can be useful for other graph execution models [35].

A. Privacy Budget Allocation

To preserve DP, a privacy budget ϵ is given to the entire graph application. The budget is further allocated to each inter-DC message for differentially private perturbation. Recall that, the larger the budget, the lower the noise. According to the composition property of DP, we first distribute ϵ evenly among multiple graph processing iterations. Second, we evenly distribute the budget of iteration i (denoted ϵ_i) to each DC. Finally, we further distribute ϵ_i to each outgoing message that needs protection. For PageRank, HITS and SALSA, the sequential composition property is satisfied among the messages and thus we evenly distribute ϵ_i among them. For ALS where parallel composition is satisfied among the messages, the budget allocated to each of them equals to ϵ_i.

B. Sampling Technique

In large-scale graph processing, it is not always necessary to transfer every message between vertices to obtain good accuracy. For example, in PageRank, the rank value of a vertex is updated using the rank of its neighbors. Thus, dropping a few messages from low-degree neighbors does not significantly reduce the accuracy. Our experiments using PageRank and LiveJournal graph on five DCs (detailed setup in Section V) show that, using random sampling to reduce the number of inter-DC messages can greatly reduce the WAN usage with good accuracy, as shown in Figure 4. Further, as we will prove later, sampling the edges amplifies the privacy budget and leads to satisfying ϵ-edge-DP with less perturbation. Thus, it can benefit both our low-latency and high-utility goals.

Given a graph application, we sample its inter-DC data communications with a sampling probability p. Users can empirically decide or smartly learn the best sampling probability according to the trade-off between graph processing accuracy and WAN usage. Although sampling only inter-DC messages may introduce skewness to graph processing results, Figure 2 shows that inter-DC messages are dominating in real geo-distributed graph applications. As a result, sampling only inter-DC messages does not affect much the accuracy. What's more, the sampling technique can help improve the accuracy of graph processing with its *amplification effect* on the privacy budget.

The amplification effect of sampling on DP was first sketched and proved for the case when the privacy budget is 1 [48]. To study the amplification effect in more general cases, we prove Lemma 1, which expands the privacy budget ϵ from 1 to an arbitrary value. In the following, A is a random algorithm which takes graphs as input.

Definition 2 (Algorithm A_p). *Let $p \in]0,1[$ and A be a random function that satisfies ϵ-DP and $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph. We define $\mathcal{E}_\mathcal{T}$ as a subset of $\mathcal{E}$ obtained by sampling each element of $\mathcal{E}$ independently with probability p (for all $x \in \mathcal{E}$, $Pr(x \in \mathcal{E}_\mathcal{T}) = p$)). If $\mathcal{G}_\mathcal{T} = (\mathcal{V}, \mathcal{E}_\mathcal{T})$ then we define $A_p(\mathcal{G}) = A(\mathcal{G}_\mathcal{T})$.*

Lemma 1 (Amplification via sampling). *If A satisfies ϵ-DP, then for any $p \in]0,1[$, A_p satisfies $p(e^\epsilon - 1)$-DP.*

Proof. In the following, let us denote as δ the function that samples any subset with independent sampling probability p for any element of the input set (using the notation from Definition 2, $\mathcal{E}_\mathcal{T} = \delta(\mathcal{E})$). For a given graph $\mathcal{G}$, we denote as $\mathcal{G}_\delta$ its subgraph after the sampling of its edge set $A_p(\mathcal{G}) = A(\mathcal{G}_\delta)$). Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ be two adjacent graphs. We assume here that $\mathcal{E} = \mathcal{E}' \cup \{x\}$.

Let now $\mathcal{E}'_\mathcal{T}$ be any subset of $\mathcal{E}'$ ($\mathcal{E}'_\mathcal{T}$ is a possible output of $\delta(\mathcal{E}')$). Let us now denote $\mathcal{E}_\mathcal{T} = \mathcal{E}'_\mathcal{T} \cup \{x\}$. Because the sampling of each element is independent, we have $Pr(\delta(\mathcal{E}) = \mathcal{E}_\mathcal{T}) = p \cdot Pr(\delta(\mathcal{E}') = \mathcal{E}'_\mathcal{T})$ and $Pr(\delta(\mathcal{E}) = \mathcal{E}'_\mathcal{T}) = (1-p) \cdot Pr(\delta(\mathcal{E}') = \mathcal{E}'_\mathcal{T})$. Thus,

$$\begin{aligned} Pr(\delta(\mathcal{E}) = \delta(\mathcal{E}')|\delta(\mathcal{E}') = \mathcal{E}'_\mathcal{T}) &= \frac{Pr(\delta(\mathcal{E}) = \mathcal{E}'_\mathcal{T} \& \delta(\mathcal{E}') = \mathcal{E}'_\mathcal{T})}{Pr(\delta(\mathcal{E}') = \mathcal{E}'_\mathcal{T})} \\ &= \frac{(1-p) \cdot Pr(\delta(\mathcal{E}') = \mathcal{E}'_\mathcal{T})^2}{Pr(\delta(\mathcal{E}') = \mathcal{E}'_\mathcal{T})} \\ &= (1-p) \cdot Pr(\delta(\mathcal{E}') = \mathcal{E}'_\mathcal{T}). \end{aligned}$$

Similarly, $Pr(\delta(\mathcal{E}) \neq \delta(\mathcal{E}')|\delta(\mathcal{E}') = \mathcal{E}'_\mathcal{T}) = p \cdot Pr(\delta(\mathcal{E}') = \mathcal{E}'_\mathcal{T})$. Note that when the sampling event "$\delta(\mathcal{E}) \neq \delta(\mathcal{E}')$" occurs, under the assumption that $\delta(\mathcal{E}') = \mathcal{E}'_\mathcal{T}$, we know that the two induced subgraphs $\mathcal{G}_\delta$ and $\mathcal{G}'_\delta$ are adjacent (because the only possible difference is the edge x being part of $\delta(\mathcal{E})$).

Let us now consider any $S \subset Im(A)$. By definition, $A_p(\mathcal{G}) \in S$ if and only if $A(\mathcal{G}_\delta) \in S$. The same applies for $A_p(\mathcal{G}')$ and $A(\mathcal{G}'_\delta)$. Thus, $Pr(A_p(\mathcal{G}) \in S|\delta(\mathcal{E}') = \mathcal{E}'_\mathcal{T}) = Pr(A(\mathcal{G}_\delta) \in S)|\delta(\mathcal{E}') = \mathcal{E}'_\mathcal{T})$.

As mentioned above, if $\delta(\mathcal{E}') = \mathcal{E}'_\mathcal{T}$, then either $\mathcal{G}_\delta = \mathcal{G}'_\delta$ (with probability $(1-p)$), or $\mathcal{G}_\delta$ and $\mathcal{G}'_\delta$ are adjacent (with probability p). Thus,

$$\begin{aligned} & Pr(A(\mathcal{G}_\delta) \in S|\delta(\mathcal{E}') = \mathcal{E}'_\mathcal{T}) \\ & \quad =(1-p) \cdot Pr(A(\mathcal{G}_\delta) \in S|\delta(\mathcal{E}') = \mathcal{E}'_\mathcal{T}|\mathcal{G}_\delta = \mathcal{G}'_\delta) \\ & \qquad + p \cdot Pr(A(\mathcal{G}_\delta) \in S|\delta(\mathcal{E}') = \mathcal{E}'_\mathcal{T}|\mathcal{G}_\delta \neq \mathcal{G}'_\delta) \\ & \quad \leq (1-p) \cdot Pr(A(\mathcal{G}'_\delta) \in S)|\delta(\mathcal{E}') = \mathcal{E}'_\mathcal{T}) \\ & \qquad + pe^\epsilon \cdot Pr(A(\mathcal{G}'_\delta) \in S)|\delta(\mathcal{E}') = \mathcal{E}'_\mathcal{T}) \\ & \quad \leq (1 + p(e^\epsilon - 1)) \cdot Pr(A(\mathcal{G}'_\delta) \in S)|\delta(\mathcal{E}') = \mathcal{E}'_\mathcal{T}) \\ & \quad \leq e^{p(e^\epsilon - 1)} \cdot Pr(A(\mathcal{G}'_\delta) \in S)|\delta(\mathcal{E}') = \mathcal{E}'_\mathcal{T}). \end{aligned}$$

as $1 + t \leq e^t$ for any t.

Thus we have $Pr(A_p(\mathcal{G}) \in S|\delta(\mathcal{E}') = \mathcal{E}'_\mathcal{T}) \leq e^{p(e^\epsilon - 1)} \cdot Pr(A_p(\mathcal{G}') \in S|\delta(\mathcal{E}') = \mathcal{E}'_\mathcal{T})$. From it we can conclude

$$\begin{aligned}
Pr(A_p(\mathcal{G}) \in S) = & \sum_{\mathcal{E}'_{\mathcal{T}} \subset \mathcal{E}'} Pr(A_p(\mathcal{G}) \in S \bigcap \delta(\mathcal{E}') = \mathcal{E}'_{\mathcal{T}}) \\
= & \sum_{\mathcal{E}'_{\mathcal{T}} \subset \mathcal{E}'} Pr(A_p(\mathcal{G}) \in S | \delta(\mathcal{E}') = \mathcal{E}'_{\mathcal{T}}) \\
& \times Pr(\delta(\mathcal{E}') = \mathcal{E}'_{\mathcal{T}}) \\
\leq & \sum_{\mathcal{E}'_{\mathcal{T}} \subset \mathcal{E}'} e^{p(e^{\epsilon}-1)} \cdot Pr(A_p(\mathcal{G}') \in S | \delta(\mathcal{E}') = \mathcal{E}'_{\mathcal{T}}) \\
& \times Pr(\delta(\mathcal{E}') = \mathcal{E}'_{\mathcal{T}}) \\
\leq & e^{p(e^{\epsilon}-1)} \cdot \sum_{\mathcal{E}'_{\mathcal{T}} \subset \mathcal{E}'} Pr(A_p(\mathcal{G}') \in S | \delta(\mathcal{E}') = \mathcal{E}'_{\mathcal{T}}) \\
& \times Pr(\delta(\mathcal{E}') = \mathcal{E}'_{\mathcal{T}}) \\
\leq & e^{p(e^{\epsilon}-1)} \cdot Pr(A_p(\mathcal{G}') \in S)
\end{aligned}$$

which proves that A_p satisfies $p(e^{\epsilon}-1)$-DP when $\mathcal{E} = \mathcal{E}' \cup \{x\}$. In the other case, the same proof applies, as the probability of $\mathcal{G}_\delta$ and $\mathcal{G}'_\delta$ being equals or adjacent are unchanged (but this time under the assumption $\delta(\mathcal{E}) = \mathcal{E}_{\mathcal{T}}$). □

We prove Lemma 2, which shows that the amplification effect of the sampling technique allows us to use a larger budget to generate noises while preserving the same level of differential privacy. The proof is straightforward.

Lemma 2. *For any $k > 0$, if A satisfies $\ln(k+1)$-DP and $\epsilon \in]0, k[$, then $A_{\epsilon/k}$ satisfies ϵ-DP.*

This is to say, if we sample the inter-DC messages with probability $p = \epsilon/k$, the original ϵ-DP will still be satisfied even if we use a bigger privacy budget, $\ln(k+1)$, for the sampled messages. When p is small, there are many dropped messages which may harm the accuracy. However, a smaller p leads to a larger privacy budget and hence compensates the accuracy loss. As a result, we can reduce inter-DC data communication using sampling without sacrificing accuracy.

C. Combiners

Similar to Pregel, we adopt combiners at each DC to combine outgoing messages. The difference is that, combiners in GeoPregel are defined for pairs of DCs that are communicating. The function of each combiner can be defined by users according to the computation of graph algorithms. For example, as shown in Figure 3b, the combine operation for PageRank is to 1) sum up the rank values of all combined messages and 2) combine the receivers of all messages into a list. On receiving a combined message, the target DC distributes the message to each receiver in the list. The distribute operation is also application-dependent. For example, for PageRank, the combined message is evenly distributed to each receiver.

The main motivation of introducing combiners in GeoPregel is to enlarge the privacy budget allocated to inter-DC messages and hence improve system utility. We define *perturbed combiners* which only combine messages that need privacy protection. According to Section IV-A, we set one perturbed combiner for each pair of communicating DCs to have the best benefit from the available privacy budget.

We can also combine inter-DC messages that do not need protection using *non-perturbed combiners* to reduce WAN usage and hence system latency. However, doing so comes with the cost of accuracy loss. To study the impact of this

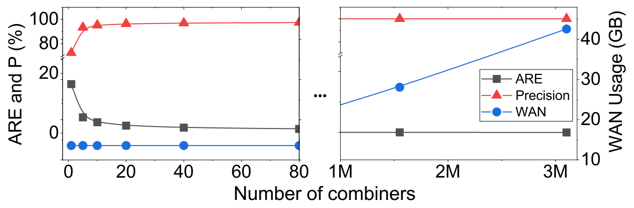


Fig. 5 Impact of non-perturbed combiners on graph processing results using PageRank algorithm and LiveJournal graph.

parameter, we vary the number of non-perturbed combiners between each pair of DCs from 1 to the number of total messages. Messages with close values are assigned to the same combiner to minimize the error introduced by combined data distribution. Privacy budget is set extremely large to only show the impact of combiners. Figure 5 shows the results of the study using PageRank algorithm and LiveJournal graph. The accuracy of processing results improves when the number of combiners increases from 1 to 15, while the WAN usage remains almost the same. When the number of combiners is 15, the ARE is almost zero and the precision is close to 100%. When the number of combiners further increases, the accuracy does not change much while the WAN usage increases significantly. Similar observations are also obtained on the other three applications. Thus, we set the number of combiners to a relatively small number to achieve both good accuracy and low latency. By default, we set the number of non-perturbed combiners to one between each pair of communicating DCs.

D. Discussion on Limitation

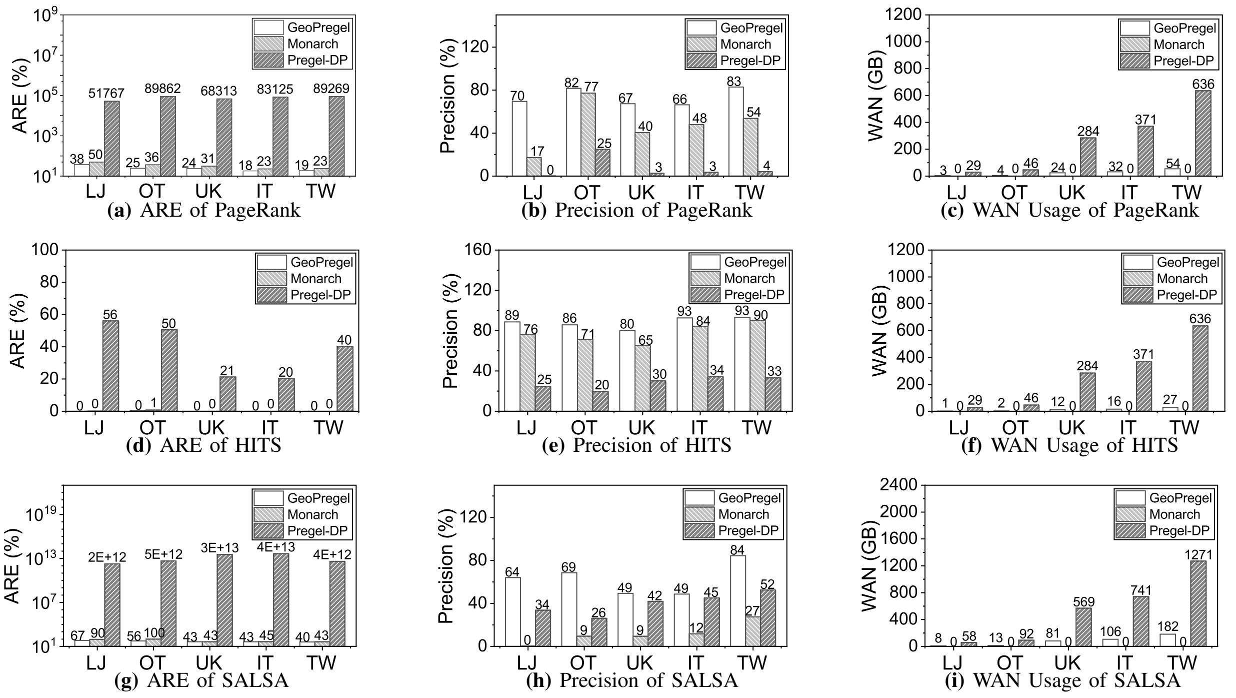
We treat the sampling, combiners and differential privacy model as building blocks to our system. When treating different graph algorithms, these building blocks can be adaptively modified accordingly. For example, for graph algorithms sending scalar values, the differential privacy model is the standard model and when the graph algorithms send set-valued data, we adopt differential privacy models specifically designed for this data type [49]. To simplify our discussion, in this paper, we focus on graph algorithms sending scalar-valued data, including a single scalar value (e.g., PageRank) or a vector of scalar values (e.g., ALS). The sampling probability and number of combiners are also application-dependent and can be tuned by users for better performance. To improve the usability of GeoPregel, it is our future work to design automated parameter tuning for different building blocks.

V. Evaluation

We integrate GeoPregel into Apache Giraph [50], an open-sourced implementation of the Pregel model. We also implemented a simulator in about 3000 LOC. We run experiments on both a physical and simulated geo-distributed cluster.

A. Dataset

Due to the difference between the graph applications, we adopt two sets of graphs. Table I shows five real-world social and web graphs for PageRank, HITS and SALSA. We adopt the MovieLens-1M and MovieLens-10M datasets for ALS as



(a) ARE of PageRank **(b)** Precision of PageRank **(c)** WAN Usage of PageRank

(d) ARE of HITS **(e)** Precision of HITS **(f)** WAN Usage of HITS

(g) ARE of SALSA **(h)** Precision of SALSA **(i)** WAN Usage of SALSA

Fig. 6 End-to-end evaluation results of compared systems on five real-world graphs and three applications.

TABLE I Social and web graphs for PageRank, HITS and ALS

Real Graph	**Vertices**	**Edges**	α_{in}	α_{out}
LiveJournal (LJ)	4,847,571	68,993,773	2.26	2.51
Orkut (OT)	3,072,441	117,185,083	2.83	1.93
uk-2005 (UK)	39,454,746	936,364,282	1.65	2.98
it-2004 (IT)	41,290,682	1,150,725,436	1.58	2.82
Twitter (TW)	41,652,230	1,468,365,182	1.80	2.02

TABLE II User-Movie ratings for ALS

Real Ratings	**Users**	**Movies**	**Ratings**
MovieLens-1M	6,040	3,706	1,000,209
MovieLens-10M	71,567	10,681	10,000,054

shown in Table II. We distribute the graphs on geo-distributed DCs based on the real geographical distribution of Twitter users [3]. Specifically, we first select five leading countries that have the largest numbers of Twitter users, including United States, Japan, India, Brazil and France. We then build a geo-distributed platform by selecting one Amazon EC2 cloud region in each of the five countries, including US West Oregon (USW), Asia Pacific Tokyo (TKY), Asia Pacific Mumbai (MUB), South America Sao Paulo (SPA) and Europe Paris (EUR). According to the proportions of Twitter users located in the five countries, we distribute 43%, 31%, 11%, 10%, and 5% of graph vertices to the USW, TKY, MUB, SPA and EUR DCs, respectively, for all graphs. Vertices are randomly mapped to different DCs. The privacy level of each region is derived from Figure 1, which are 2, 3, 1, 3 and 3, respectively, for USW, TKY, MUB, SPA and EUR. We adopt real network bandwidths measured between real cloud regions to evaluate graph processing performance.

B. Compared Solutions

We compare GeoPregel to the following solutions, where the first three are compared for all applications and the last one is specialized for ALS.

- **Pregel** [34] is the baseline comparison which does not consider data privacy issue. We use it as a baseline for accuracy and WAN usage measurements.
- **Pregel-DP** is Pregel integrated with standard DP techniques to ensure privacy. Specifically, Pregel-DP adopts the same budget allocation method as GeoPregel and it perturbs every cross-DC message. As we are the first to preserve DP in geo-distributed graph processing, we implement Pregel-DP as the state-of-the-art privacy-preserving graph engine to show the effectiveness of GeoPregel.
- **Monarch** [6] is designed for geo-distributed graph processing which uses local computation and global communication to reduce WAN usage and improve system efficiency. Monarch does not consider privacy issue and we use only local computations in the implementation to guarantee data privacy. Although we could also incorporate DP in Monarch to preserve privacy, this would introduce too much noise and leads to poor accuracy.
- **PALS** [51] is a DP-based solution specifically designed for ALS. It uses a bounded variant to improve prediction accuracy. As Root Mean Square Error (RMSE) is often used to measure the prediction accuracy of ALS algorithms, we adopt RMSE as the utility measure for ALS application.

For all comparisons, we run the applications with the same fixed number of iterations. We set the number to 20 for PageRank and 10 for the rest applications, according to their convergence speed when running on Pregel.

C. End-to-End Evaluations

We run a physical cluster of six nodes to emulate the geo-distributed environment, where one node works as the master

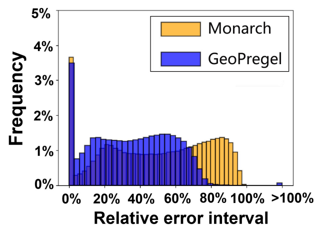


Fig. 7 Relative error distribution of a vertex in LiveJournal using PageRank algorithm.

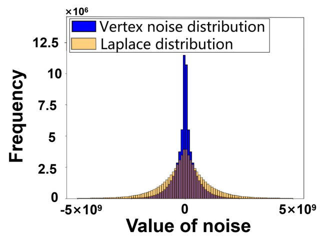
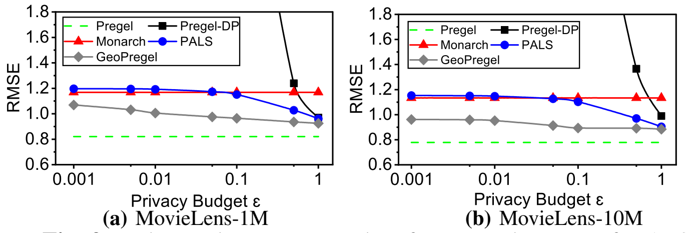
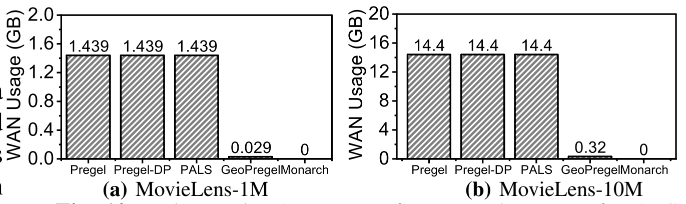


Fig. 8 Vertex noise distribution in the first iteration of SALSA on LiveJournal.



(a) MovieLens-1M **(b)** MovieLens-10M

Fig. 9 End-to-end accuracy results of compared systems for ALS.



(a) MovieLens-1M **(b)** MovieLens-10M

Fig. 10 End-to-end WAN usage of compared systems for ALS.

of Giraph and the other five are workers each representing a cloud DC. During graph processing, messages are transferred across the five workers and we control the network bandwidths between nodes using network traces measured from Amazon EC2. Below we present the end-to-end performance of GeoPregel and the comparisons.

1) Overall Results of PageRank, HITS and SALSA: Figure 6 shows how GeoPregel compares with Pregel-DP and Monarch with respect to the quality measures on the five real-world graphs for PageRank, HITS and SALSA. We set the sampling rate to 30%, 60% and 100% for HITS, PageRank and SALSA, respectively. The privacy budget is 1 and the top-k size is 2%. We have the following observations.

First, looking at the accuracy, GeoPregel obtains the lowest average relative error (ARE) and the highest precision for all applications on all graphs. The ARE of GeoPregel is 99%-100% and 1%-74% lower than Pregel-DP and Monarch, respectively, and the precision of GeoPregel is 8%-1225x and 4%-154x higher than Pregel-DP and Monarch, respectively. Taking a closer look at the results, Figure 7 shows the relative error distributions of a vertex in the LiveJournal graph when running PageRank using Monarch and GeoPregel. The results demonstrate that our proposed techniques are effective in improving the accuracy of geo-distributed graph processing while satisfying differential privacy.

Second, Pregel-DP performs the worst among the three comparisons, with extremely large error and low precision in most cases. This is mainly due to the large amount of noises added to inter-DC data communications during graph processing. The precision of Pregel-DP is relatively high for SALSA, which is due to the fact that an inter-DC message in SALSA requires adding twice the Laplace noise before being used to update the value of a target vertex. This results in a noise cancellation effect which helped to improve the accuracy of Pregel-DP. To demonstrate this effect, we show in Figure 8 the distribution of vertex noise in the first iteration of SALSA using LiveJournal graph. The noise added to each inter-DC message is sampled from the Laplace distribution shown in the figure. Clearly, the noise added to the vertices in one SALSA iteration can lead to smaller error compared to the noise sampled directly from the Laplace distribution.

Third, by avoiding all inter-DC communications, Monarch can reach good accuracy for some applications such as HITS. Monarch achieves very low ARE and high precision on HITS, which are 0-1% and 65%–90%, respectively. The reason that ARE is very low is that we normalize the hub and authority scores of a vertex to [1,2) by adding 1 to a small value in [0,1) calculated using all messages received by the vertex. Thus, the impact of missing messages on the ARE of the scores becomes less significant. However, Monarch performs poorly on the other two applications, especially on SALSA. This is again due to the fact that one vertex update in SALSA requires two message passing to get the desired data, which is more vulnerable to message dropping as in Monarch.

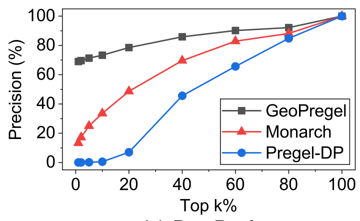
Lastly, when looking at WAN usage, GeoPregel achieves lower WAN usage than Pregel-DP at all times. Specifically, GeoPregel reduces the WAN usage by 86%-96% compared to Pregel-DP. This shows that our techniques are effective in optimizing system latency. Although Monarch leads to zero inter-DC communication, it has poor generality in obtaining good graph processing accuracy. In contrast, GeoPregel is able to obtain good accuracy on all three applications with very low inter-DC data communication overhead.

2) Overall Results of ALS: Figure 9 and 10 show the accuracy and WAN usage results of compared algorithms for the ALS application. We set the sampling probability of GeoPregel to 30%. The privacy budget varies from 0.001, 0.01, 0.1 to 1. We have the following observations.

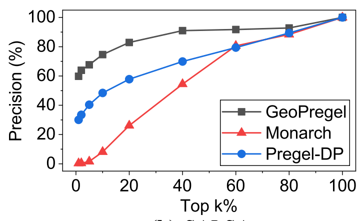
First, GeoPregel obtains the lowest RMSE results among all privacy-preserving comparisons. When the privacy budget is 1, all three DP-based solutions, namely Pregel-DP, PALS and GeoPregel, obtain equally good accuracy results. With the decrease of the budget, the RMSE result of Pregel-DP increases rapidly. GeoPregel is able to obtain better accuracy results compared to PALS, the DP-based solution specifically tailored for ALS, more obvious at low privacy budgets.

Second, similar to the first three applications, GeoPregel can greatly reduce the WAN usage and hence system latency for ALS compared to the other algorithms except Monarch. For example, GeoPregel reduces the WAN usage by 98% compared to the other two DP-based solutions (i.e., Pregel-DP and PALS). Although the WAN usage of GeoPregel is slightly higher than Monarch, it obtains much lower RMSE.

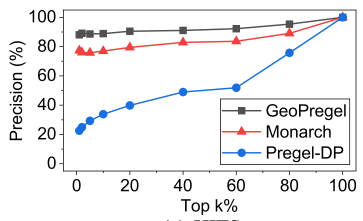
In the following, we perform a set of sensitivity studies



(a) PageRank

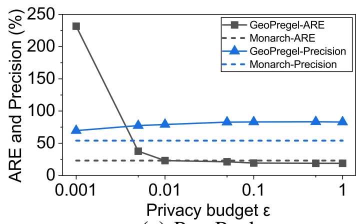


(b) SALSA

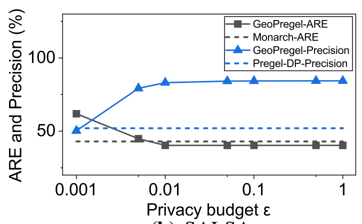


(c) HITS

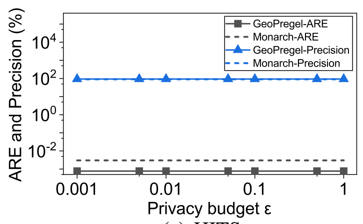
Fig. 11 Evaluating the effectiveness of GeoPregel under different top-k sizes using Twitter graph.



(a) PageRank



(b) SALSA



(c) HITS

Fig. 12 Sensitivity study on the privacy budget ϵ using Twitter graph. Dashed lines represent the ARE and precision results of Monarch and Pregel-DP (whichever performs better).

to show the effectiveness of GeoPregel. For simplicity yet without loss of generality, we only use PageRank, HITS and SALSA which adopt the same quality measures.

D. Effectiveness under Varying Top-k Sizes

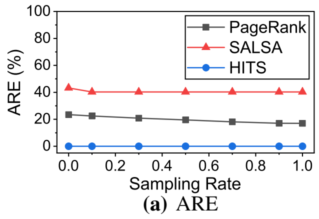
As precision is more relevant measure for recommendation-based applications, we report the precision obtained by the compared systems under different top-k sizes. Specifically, we vary k from 1%, 2%, 5%, 10% to 100% of the number of vertices using Twitter graph and show the results in Figure 11. We have the following observations.

First, GeoPregel obtains the highest precision among the compared solutions for all k values and all applications. This demonstrates the effectiveness of GeoPregel regardless of the value of k. Second, graph processing precision increases as k increases for all three comparisons. This is because when we increase the value of k, the vertices that were not in the top-k list of the baseline results may have a chance to get in the longer list. Hence the precision of the results measured using the larger k value will increase. By default, we set k to 2%.

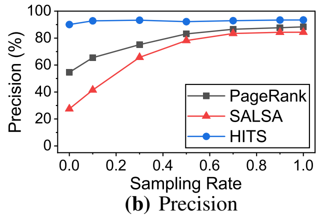
E. Impact of Privacy Budget ϵ

The degree of privacy protection is related to ϵ. The smaller the ϵ, the higher the protection and the larger the noise. We study the accuracy of GeoPregel under different ϵ using Twitter graph, so as to learn whether GeoPregel can support higher privacy protection. We vary ϵ from 0.001, 0.005, 0.01, 0.05, 0.1, 0.5 to 1. We set ϵ to 1 by default. Figure 12 shows the accuracy results of the comparisons using simulator. We have the following observations.

First, GeoPregel obtains higher precision and lower ARE when ϵ increases. This is expected as a smaller privacy budget leads to more noise added during geo-distributed graph processing. Second, GeoPregel can achieve better accuracy than state-of-the-art comparisons even at a very low budget. For example, for PageRank and SALSA, GeoPregel obtains higher precision compared to Monarch and Pregel-DP under



(a) ARE



(b) Precision

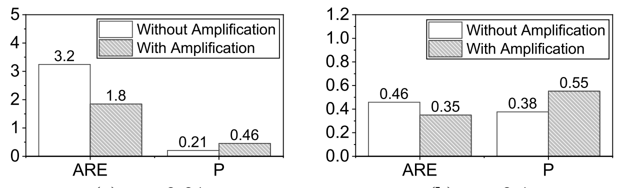
Fig. 13 Sensitivity study on the sampling rate using Twitter.

all budgets. This means GeoPregel can guarantee very tight differential privacy levels while improving graph processing accuracy. Third, GeoPregel shows different degrees of sensitivity on the ϵ parameter for different applications. For example, the accuracy of GeoPregel decreases abruptly when ϵ is lower than 0.01 for PageRank, and less so for SALSA and HITS. This is partly due to the different global sensitivities of the applications. Both SALSA and HITS have global sensitivity of 1 while PageRank has a larger bounded sensitivity. Thus, PageRank is injected with larger amounts of noise compared to the other two applications and is more vulnerable to small privacy budgets. The reason that HITS is less sensitive to changes in privacy budgets compared to SALSA is due to the normalization operation in HITS which bounds the noise to the range of (0,1).

F. Impact of Sampling Rate

The sampling technique can both positively and adversely affect the accuracy of GeoPregel due to the budget amplification effect and the loss of useful inter-DC information. We study its impact by varying the sampling rate from 0, 10%, 20%, . . . , to 100% using Twitter graph. Figure 13 shows the obtained results. Note that, when the sampling rate is 0, GeoPregel is equivalent to Monarch since there is no inter-DC communication. We have the following observations.

First, the graph processing precision obtained by GeoPregel decreases along with the decrease in sampling rate. This is



(a) $\epsilon = 0.01$ **(b)** $\epsilon = 0.1$

Fig. 14 Impact of budget amplification using Livejournal.

expected as a higher sampling rate maintains more useful information and hence leads to higher precision. The ARE results slightly increase with the decrease of the sampling rate. This is mainly due to the budget amplification effect which helps to diminish the adverse impact of message dropping on the accuracy of graph processing results.

Second, GeoPregel shows different degrees of sensitivity dependence on sampling rate for different applications. For example, the precision of GeoPregel remains high with respect to HITS (above 90%) even with decreasing sampling rates. This is consistent with the good precision results of Monarch for HITS as shown in Figure 6. Note that, the WAN usage is almost linearly related to the sampling rate. Thus, we prefer choosing a small sampling rate for HITS to achieve good accuracy and low WAN usage at the same time. PageRank is less sensitive to the sampling rate compared to SALSA. Thus, we can choose a smaller sampling rate for PageRank than that for SALSA. By default, we set the sampling rate to 30%, 60% and 100% for HITS, PageRank and SALSA, respectively, for a good balance between system utility and latency.

G. Impact of Budget Amplification

Budget amplification can effectively reduce the accuracy loss caused by DP perturbation. To clearly show the positive effects brought by budget amplification, we compared the accuracy of using sampling with and without budget amplification. We perform experiments using simulator and set the privacy budget to 0.01 and 0.1 for Livejournal graph and PageRank algorithm. Figure 14 shows the obtained results. We have the following observations.

First, the budget amplification effect is useful on improving the accuracy of graph processing results under different privacy budgets. Specifically, when ϵ is 0.01 and 0.1, the budget amplification reduces the ARE by 44% and 24%, respectively, and improves the precision by 119% and 45%, respectively. Second, the budget amplification effect is more useful on improving the accuracy of graph processing results when the privacy budget is lower. This shows that GeoPregel can achieve good utility even under strict privacy protection requirements.

VI. Related Work

A. Geo-Distributed Graph Processing

There have been many graph processing systems [52], [53], [54], [55], [56], [57] proposed for scalable and efficient graph executions in single DCs. Recently, there have been some studies focusing on the cost and performance optimizations for graph processing in geo-distributed DCs via graph partitioning [4], [5], [58]. However, such studies require vertex migration to perform graph partitioning, which introduces large inter-DC data movement and may violate privacy regulations when used in geo-distributed environments.

A few graph execution models have been proposed specifically for geo-distributed DCs [59], [6], [7], [60]. For example, Monarch [6] proposed to use sampling technique and incremental computation to reduce the data exchanged across geo-distributed DCs. HSP [59] is extended from the BSP model by performing synchronization in a two-level hierarchy to get a lower WAN bandwidth usage and faster convergence. GeoGraph [7] is a universal framework to support efficient geo-distributed graph query processing based on clustering DCs and meta-graph. None of the studies mentioned above has considered the privacy issue in geo-distributed DCs, which is crucial to big data applications [11].

B. Privacy-Preserving Graph Processing

The privacy issue has attracted great attention from different applications. Quoc et. al [12] proposed a privacy-preserving stream analytics system on distributed users' private dataset. Agarwal et. al [13] proposed a privacy-preserving model for distributed machine learning based on DP. However, these systems cannot be easily adapted to graph applications which have different computation and communication patterns. Privacy-preserving graph publishing techniques have been widely studied to preserve the utility of graphs while preserving graph data privacy [14], [15], [16], [17], [18]. Although such studies can protect graph structures, there still lacks an end-to-end solution which can preserve graph data privacy during geo-distributed execution without modifying the applications.

Achieving privacy preservation during graph processing is non-trivial. Although existing techniques such as data encryption [19], [20], [21] and secure multi-party computation [22], [23] can be used to preserve data privacy, they are known for high latency. Differential privacy (DP) [24] is a general and lightweight technique for privacy-preserving and has been explored by existing studies [61], [62], [63], [64] in collaborative filtering recommendation systems to protect personal data privacy. Sealfon [65] uses DP when calculating the shortest path of graphs, where the graph topology is public and the private information consists only of edge weights. However, these studies are proposed for a specific graph application only. To the best of our knowledge, there is no universal DP solution that can adapt to different graph applications.

VII. Conclusion

In this paper, we propose *GeoPregel*, an end-to-end system for privacy-preserving graph processing in geo-distributed DCs with *high utility* and *low latency*. GeoPregel adopts differential privacy (DP) to preserve privacy and incorporates two techniques including sampling and combiners to reduce the impact of DP perturbation hence improving the accuracy of graph processing results. The two techniques are also useful for reducing inter-DC data communication and hence reducing system latency. Evaluations using real-world graphs and real cloud traces have demonstrated the effectiveness of GeoPregel. We have open-sourced GeoPregel.

References

[1] A. Ching, S. Edunov, M. Kabiljo, D. Logothetis, and S. Muthukrishnan, "One trillion edges: Graph processing at facebook-scale," *Proceedings of the VLDB Endowment*, vol. 8, no. 12, pp. 1804–1815, 2015.

[2] P. Vagata and K. Wilfong, *Scaling the Facebook data warehouse to 300 PB*, 2014, https://bit.ly/3qPEG2D.

[3] Statista, *Leading countries based on number of Twitter users as of July 2021*, https://bit.ly/32IM2NC.

[4] A. C. Zhou, B. Shen, Y. Xiao, S. Ibrahim, and B. He, "Cost-aware partitioning for efficient large graph processing in geo-distributed datacenters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 7, pp. 1707–1723, 2019.

[5] A. C. Zhou, S. Ibrahim, and B. He, "On achieving efficient data transfer for graph processing in geo-distributed datacenters," in *2017 IEEE 37th international conference on distributed computing systems (ICDCS)*. IEEE, 2017, pp. 1397–1407.

[6] A. P. Iyer, A. Panda, M. Chowdhury, A. Akella, S. Shenker, and I. Stoica, "Monarch: gaining command on geo-distributed graph analytics," in *10th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 18)*, 2018.

[7] Y. Yuan, D. Ma, Z. Wen, Y. Ma, G. Wang, and L. Chen, "Efficient graph query processing over geo-distributed datacenters," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 619–628.

[8] Forrester Research, "Forrester's Global Map of Privacy Rights and Regulations," https://bit.ly/3mXl9Mw, 2021.

[9] A. Deshpande and A. Machanavajjhala, "Privacy challenges in the post-gdpr world: A data management perspective," https://bit.ly/3zuiyP7, 2018.

[10] E. Commission, "Article 45 gdpr: transfers on the basis of an adequacy decision," https://bit.ly/3qPD9cT, 2016.

[11] F. T. Commission, "FTC Imposes $5 Billion Penalty and Sweeping New Privacy Restrictions on Facebook," https://bit.ly/3eRUwnP, 2019.

[12] D. L. Quoc, M. Beck, P. Bhatotia, R. Chen, C. Fetzer, and T. Strufe, "PrivApprox: Privacy-Preserving stream analytics," in *2017 USENIX Annual Technical Conference (USENIX ATC 17)*. Santa Clara, CA: USENIX Association, Jul. 2017, pp. 659–672. [Online]. Available: https://bit.ly/3pRMaD4

[13] N. Agarwal, A. T. Suresh, F. Yu, S. Kumar, and H. B. McMahan, "Cpsgd: Communication-efficient and differentially-private distributed sgd," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, p. 7575–7586.

[14] X.-Y. Li, C. Zhang, T. Jung, J. Qian, and L. Chen, "Graph-based privacy-preserving data publication," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.

[15] X. Ding, C. Wang, K.-K. R. Choo, and H. Jin, "A novel privacy preserving framework for large scale graph data publishing," *IEEE transactions on knowledge and data engineering*, vol. 33, no. 2, pp. 331–343, 2019.

[16] Q. Li, J. S. Gundersen, R. Heusdens, and M. G. Christensen, "Privacy-preserving distributed processing: Metrics, bounds and algorithms," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2090–2103, 2021.

[17] M. Hay, C. Li, G. Miklau, and D. D. Jensen, "Accurate estimation of the degree distribution of private networks," *2009 Ninth IEEE International Conference on Data Mining*, pp. 169–178, 2009.

[18] Z. Jorgensen, T. Yu, and G. Cormode, "Publishing attributed social graphs with formal privacy guarantees," *Proceedings of the 2016 International Conference on Management of Data*, 2016.

[19] C. Zhang, L. Zhu, C. Xu, K. Sharif, C. Zhang, and X. Liu, "Pgas: Privacy-preserving graph encryption for accurate constrained shortest distance queries," *Information Sciences*, vol. 506, pp. 325–345, 2020.

[20] E. Ghosh, S. Kamara, and R. Tamassia, "Efficient graph encryption scheme for shortest path queries," in *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, 2021, pp. 516–525.

[21] A. Viand, P. Jattke, and A. Hithnawi, "Sok: Fully homomorphic encryption compilers," *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 1092–1108, 2021.

[22] O. Goldreich, "Foundations of cryptography: Volume 2, basic applications," 2004.

[23] D. E. Evans, V. Kolesnikov, and M. Rosulek, "A pragmatic introduction to secure multi-party computation," *Found. Trends Priv. Secur.*, vol. 2, pp. 70–246, 2018.

[24] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, pp. 211–407, 2014.

[25] N. Li, W. Qardaji, and D. Su, "On sampling, anonymization, and differential privacy or, k-anonymization meets differential privacy," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, 2012, pp. 32–33.

[26] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith, "What can we learn privately?" *SIAM Journal on Computing*, vol. 40, no. 3, pp. 793–826, 2011.

[27] A. Beimel, S. P. Kasiviswanathan, and K. Nissim, "Bounds on the sample complexity for private learning and private data release," in *Theory of Cryptography Conference*. Springer, 2010, pp. 437–454.

[28] Y.-X. Wang, S. Fienberg, and A. Smola, "Privacy for free: Posterior sampling and stochastic gradient monte carlo," in *International Conference on Machine Learning*. PMLR, 2015, pp. 2493–2502.

[29] M. Bun, C. Dwork, G. N. Rothblum, and T. Steinke, "Composable and versatile privacy via truncated cdp," in *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, 2018, pp. 74–86.

[30] Y.-X. Wang, B. Balle, and S. P. Kasiviswanathan, "Subsampled rényi differential privacy and analytical moments accountant," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 1226–1235.

[31] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 308–318.

[32] C. Eksombatchai, P. Jindal, J. Z. Liu, Y. Liu, R. Sharma, C. Sugnet, M. Ulrich, and J. Leskovec, "Pixie: A system for recommending 3+ billion items to 200+ million users in real-time," in *Proceedings of the 2018 world wide web conference*, 2018, pp. 1775–1784.

[33] A. Sharma, J. Jiang, P. Bommannavar, B. Larson, and J. Lin, "Graphjet: Real-time content recommendations at twitter," *Proceedings of the VLDB Endowment*, vol. 9, no. 13, pp. 1281–1292, 2016.

[34] 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 *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, 2010, pp. 135–146.

[35] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin, "Powergraph: Distributed graph-parallel computation on natural graphs," in *10th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 12)*, 2012, pp. 17–30.

[36] H. Kwak, C. Lee, H. Park, and S. Moon, "What is Twitter, a social network or a news media?" in *WWW '10*, 2010, pp. 591–600.

[37] Z. Brakerski, N. Döttling, S. Garg, and G. Malavolta, "Candidate io from homomorphic encryption schemes," 2020.

[38] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Tfhe: fast fully homomorphic encryption over the torus," *Journal of Cryptology*, vol. 33, no. 1, pp. 34–91, 2020.

[39] J. Li, X. Kuang, S. Lin, X. Ma, and Y. Tang, "Privacy preservation for machine learning training and classification based on homomorphic encryption schemes," *Information Sciences*, vol. 526, pp. 166–179, 2020.

[40] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.

[41] J. M. Kleinberg *et al.*, "Authoritative sources in a hyperlinked environment." in *SODA*, vol. 98. Citeseer, 1998, pp. 668–677.

[42] R. Lempel and S. Moran, "Salsa: the stochastic approach for link-structure analysis," *ACM Transactions on Information Systems (TOIS)*, vol. 19, no. 2, pp. 131–160, 2001.

[43] D. Zachariah, M. Sundin, M. Jansson, and S. Chatterjee, "Alternating least-squares for low-rank matrix reconstruction," *IEEE Signal Processing Letters*, vol. 19, no. 4, pp. 231–234, 2012.

[44] M. A. Najork, "Comparing the effectiveness of hits and salsa," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, 2007, pp. 157–164.

[45] P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang, and R. Zadeh, "Wtf: The who to follow service at twitter," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 505–514.

[46] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of cryptography conference*. Springer, 2006, pp. 265–284.

[47] F. D. McSherry, “Privacy integrated queries: An extensible platform for privacy-preserving data analysis,” in *Proc. of ACM SIGMOD*, 2009. [Online]. Available: http://doi.acm.org/10.1145/1559845.1559850

[48] A. Smith, “Differential privacy and the secrecy of the sample,” https://bit.ly/3nbYU5T, 2009.

[49] R. Chen, N. Mohammed, B. C. Fung, B. C. Desai, and L. Xiong, “Publishing set-valued data via differential privacy,” *Proceedings of the VLDB Endowment*, vol. 4, no. 11, pp. 1087–1098, 2011.

[50] Apache, “Apache Giraph,” https://giraph.apache.org/, 2020.

[51] A. Friedman, S. Berkovsky, and M. A. Kaafar, “A differential privacy framework for matrix factorization recommender systems,” *User Modeling and User-Adapted Interaction*, vol. 26, no. 5, pp. 425–458, 2016.

[52] Z. Ai, M. Zhang, Y. Wu, X. Qian, K. Chen, and W. Zheng, “Squeezing out all the value of loaded data: An out-of-core graph processing system with reduced disk i/o,” in *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, 2017, pp. 125–137.

[53] R. Chen, J. Shi, Y. Chen, B. Zang, H. Guan, and H. Chen, “Powerlyra: Differentiated graph computation and partitioning on skewed graphs,” *ACM Transactions on Parallel Computing (TOPC)*, vol. 5, no. 3, pp. 1–39, 2019.

[54] S. Grossman, H. Litz, and C. Kozyrakis, “Making pull-based graph processing performant,” *ACM SIGPLAN Notices*, vol. 53, no. 1, pp. 246–260, 2018.

[55] W. Xie, G. Wang, D. Bindel, A. Demers, and J. Gehrke, “Fast iterative graph computation with block updates,” *Proceedings of the VLDB Endowment*, vol. 6, no. 14, pp. 2014–2025, 2013.

[56] M. Zhang, Y. Zhuo, C. Wang, M. Gao, Y. Wu, K. Chen, C. Kozyrakis, and X. Qian, “Graphp: Reducing communication for pim-based graph processing with efficient data partition,” in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2018, pp. 544–557.

[57] X. Xie, X. Wei, R. Chen, and H. Chen, “Pragh: Locality-preserving graph traversal with split live migration,” in *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, 2019, pp. 723–738.

[58] H. Li, H. Yuan, J. Huang, J. Cui, X. Ma, S. Wang, J. Yoo, and S. Y. Philip, “Group reassignment for dynamic edge partitioning,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 10, pp. 2477–2490, 2021.

[59] S. Liu, L. Chen, B. Li, and A. Carnegie, “A hierarchical synchronous parallel model for wide-area graph analytics,” in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 531–539.

[60] K. Beedkar, J.-A. Quiané-Ruiz, and V. Markl, “Compliant geo-distributed query processing,” in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 181–193.

[61] A. Machanavajjhala, A. Korolova, and A. D. Sarma, “Personalized social recommendations-accurate or private?” *arXiv preprint arXiv:1105.4254*, 2011.

[62] F. McSherry and I. Mironov, “Differentially private recommender systems: Building privacy into the netflix prize contenders,” in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 627–636.

[63] A. Berlioz, A. Friedman, M. A. Kaafar, R. Boreli, and S. Berkovsky, “Applying differential privacy to matrix factorization,” in *Proceedings of the 9th ACM Conference on Recommender Systems*, 2015, pp. 107–114.

[64] Z. Xian, Q. Li, G. Li, and L. Li, “New collaborative filtering algorithms based on svd++ and differential privacy,” *Mathematical Problems in Engineering*, vol. 2017, 2017.

[65] A. Sealfon, “Shortest paths and distances with differential privacy,” in *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, 2016, pp. 29–41.

[66] B. Xiang, Q. Liu, E. Chen, H. Xiong, Y. Zheng, and Y. Yang, “Pagerank with priors: An influence propagation perspective,” in *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[67] Q. Liu, B. Xiang, N. J. Yuan, E. Chen, H. Xiong, Y. Zheng, and Y. Yang, “An influence propagation view of pagerank,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 11, no. 3, pp. 1–30, 2017.

[68] A. Cheng and E. Friedman, “Manipulability of pagerank under sybil strategies,” 2006.

APPENDIX

A. Targeted Graph Applications

Recommendation systems are an important type of applications that often have data spread in multiple geo-distributed data centers. In this work, we target three graph algorithms that are widely adopted in recommendation systems, including PageRank [40], HITS [41] and SALSA [42].

1) PageRank: The PageRank algorithm was originally proposed by Google to measure the importance of different webpages according to the link relationship between the webpages.

Algorithm 1 presents the pseudocode of PageRank under the Pregel model. PageRank is executed iteratively in a sequence of supersteps. In each superstep, each vertex updates its rank value according to the messages received from its neighbors in the last superstep, where a message is the neighbor's pagerank divided by the number of outgoing edges. Let $PR(u)$ be the rank value of vertex u and $N_{in}(u)$ be the set of incoming neighbors of u. In the compute stage, the rank value is updated as $PR(u) = \sum_{v \in N_{in}(u)} M(v)$, where $M(v)$ represents the message (i.e., rank value) sent from vertex v to vertex u during the last superstep. Each vertex then sends out an equal share of its new rank value, $PR(u)$, to each of its outgoing neighbors. Let $N_{out}(u)$ be the set of outgoing neighbors of u. Then the value sent on each of the outgoing messages is $M(u) = PR(u)/|N_{out}(u)|$.

Algorithm 1: PageRank

```
Input: $\mathcal{G}$: the graph dataset; $K$: max number of supersteps;
foreach $k = 0$ to $K$ do
    foreach vertex $u$ in $\mathcal{G}$ do
        $PR(u) = \sum_{v \in N_{in}(u)} M(v)$;
        $M(u) = PR(u)/|N_{out}(u)|$;
        Send $M(u)$ to all $N_{out}(u)$;
```

2) HITS: HITS is another algorithm used to rank webpages relevant for a topic search. Unlike PageRank, HITS divides webpages into two types known as *hubs* and *authorities*. Given a topic search, the set of highly relevant webpages are called **Authorities** and webpages that are not very relevant but point to many related authorities are called **Hubs**. Algorithm 2 presents the flow of HITS under the Pregel model.

The first step of HITS is to create a focused subgraph using the webpages in the base set B_δ and all links among those pages. The HITS algorithm is executed on this subgraph to obtain authority and hub scores for each vertex. In this work, without loss of generality, we select all vertices of $\mathcal{G}$ as roots so that B_δ is actually equivalent to $\mathcal{G}$. Initially, the hub and authority scores of all vertices are set as 1. In each superstep, we update the two values in a mutual recursion. An authority score is computed as the sum of the scaled hub scores that point to that page. That is, $a_u = \sum_{v \in N_{in}(u)} M_h(v)$, where a_u is the authority score of vertex u and $M_h(v)$ is the hub value sent from vertex v. A hub score is the sum of the scaled authority scores of the pages it points to. That is, $h_u = \sum_{v \in N_{out}(u)} M_a(v)$, where h_u is the hub score of vertex u and $M_a(v)$ is the authority value sent from vertex v. The computed scores are then normalized to ensure convergence. Each vertex then sends out the normalized authority score to all its incoming neighbors and hub score to all its outgoing neighbors. We normalize the scores to $[1, 2]$ instead of $[0, 1]$ as in the original algorithm [41]. By doing so, we can achieve faster convergence while keeping the same ranking results of the algorithm.

Algorithm 2: HITS

```
Input: $\mathcal{G}$: the graph dataset; $K$: max number of supersteps;
Put all vertices into the root set $R_\delta$;
Expand all neighbors of vertices in $R_\delta$ into a base set $B_\delta$;
Let $a_u$ and $h_u$ be the authority and hub scores of vertex $u$;
For $\forall u \in B_\delta$, let $a_u = h_u = 1$;
foreach $k = 0$ to $K$ do
    foreach vertex $u$ in $B_\delta$ do
        if $k! = 0$ then
            $a_u = \sum_{v \in N_{in}(u)} M_h(v)$ and
            $h_u = \sum_{v \in N_{out}(u)} M_a(v)$;
        $M_a(u) = \frac{a_u}{\sqrt{\sum_{v \in B_\delta} a_v^2}} + 1$ and $a_u = M_a(u)$;
        $M_h(u) = \frac{h_u}{\sqrt{\sum_{v \in B_\delta} h_v^2}} + 1$ and $h_u = M_h(u)$;
        Send $M_a(u)$ to all $N_{in}(u)$ and $M_h(u)$ to all
          $N_{out}(u)$;
```

3) SALSA: SALSA is a webpage ranking algorithm inspired by PageRank and HITS. It has been shown to be one of the most effective link analysis algorithms [44] and has been applied in social networks such as Twitter [45] to suggest accounts to follow.

Algorithm 3 presents the flow of SALSA under the Pregel model. Similar to HITS, SALSA divides webpages into hubs and authorities and constructs a bipartite graph by putting hubs on one side and authorities on the other. It has two main differences from HITS. First, the authority/hub score of each vertex in SALSA is determined by the authority/hub scores of other vertices, while HITS uses "mutual enforcement" to update those values. Second, SALSA updates the authority/hub scores similar to the PageRank algorithm, by performing two independent random walks (i.e., a hub walk and an authority walk) on the neighborhood graph.

4) ALS: ALS is a popular matrix factorization algorithm that tries to make item recommendations to users given sparse user-item ratings. Essentially, ALS decomposes a rating matrix into the product of two lower dimensional matrices to capture the potential factors of users and items. The two matrices are alternatively updated in multiple iterations until convergence. Algorithm 4 presents the flow of ALS under the Pregel model.

Algorithm 3: SALSA

```
Put all vertices into the root set $R_\delta$;
Expand all neighbors of vertices in $R_\delta$ into a base set $B_\delta$;
Let $a_u$ and $h_u$ be the authority and hub scores of vertex $u$;
For $\forall u \in B_\delta$, let $a_u = h_u = 1$;
foreach $k = 0$ to $K$ do
    foreach vertex $u$ in $B_\delta$ do
        if $k! = 0$ then
            $a'_u = \sum_{m \in M_{in}(u)} m$ and $h'_u = \sum_{m \in M_{out}(u)} m$;
        else
            $a'_u = h'_u = 1$;
        if $k\%2 == 0$ then
            Update: $a_u = a'_u$, $h_u = h'_u$;
        foreach $v \in N_{out}(u)$ do
            $M_{in}(v).push(\frac{h'_u}{|N_{out}(u)|})$;
        foreach $v \in N_{in}(u)$ do
            $M_{out}(v).push(\frac{a'_u}{|N_{in}(u)|})$;
```

Algorithm 4: ALS

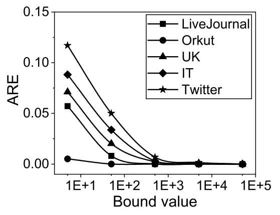
```
Let $p_u$ and $q_i$ be the vector of user $u$ and item $i$, respectively.
Let $\forall p_u$ and $\forall q_i$ be matrices with the random number;
Input: $\mathcal{G}$: the user ratings;
$K$: max number of supersteps;
$d$: number of factors;
$\lambda$: regularization parameter;
Output: Approximate factor matrices $P_{n \times d}$ and $Q_{m \times d}$
foreach $k = 0$ to $K$ do
    foreach each user u, given $Q$ do
        $p_u \leftarrow argmin_{p_u} J_Q(p_u, R_u)$
    foreach each item i, given $P$ do
        $q_i \leftarrow argmin_{q_i} J_P(q_i, R_i)$
    Send $\forall p_u$ to all $N_{in}(u)$ and $\forall q_i$ to all $N_{out}(i)$;
```

B. Global Sensitivities of Applications

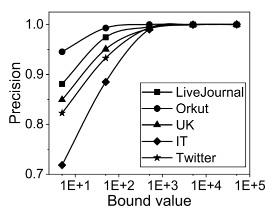
We calculate below the respective global sensitivity of the three targeted graph applications. Denote $\mathcal{G}_1$ and $\mathcal{G}_2$ as two neighboring graph partitions.

1) PageRank: For PageRank, the largest possible difference of the rank value between any pair of neighboring graph partitions is upper-bounded by the difference between the maximum and the minimum rank values, namely $\max \|PageRank(\mathcal{G}_1) - PageRank(\mathcal{G}_2)\|_1 < (pr_{max} - pr_{min})$, where pr_{max} (resp. pr_{min}) is the maximum (resp. minimum) rank value. In theory, the maximum rank value is unbounded, leading to an infinite global sensitivity. However in practice, a reasonable maximum upper bound can be set [66], [67], [68]. We experimentally show in Figure 15 using Pregel and five real-world graphs that choosing reasonable upper bounds leads to a bounded sensitivity without suffering from a dramatic utility loss (detailed experimental setup can be found in the evaluation section).

2) HITS: Since the results of HITS are normalized, the largest possible difference between any pair of neighboring



(a) ARE of bounded PageRank



(b) P of bounded PageRank

Fig. 15 Impact of upper bound on the accuracy of PageRank.

TABLE III The L_2-sensitivity of updates to p_u and q_i

Sensitivity	Bounded	Unbounded
p_u	$4q_{max} \cdot B$	$2q_{max} \cdot (B + q_{max} \cdot p_{max}) + 2\lambda p_{max}$
q_i	$4p_{max} \cdot B$	$2p_{max} \cdot (B + q_{max} \cdot p_{max}) + 2\lambda q_{max}$

graph partitions is lower-bounded by 1 and upper-bounded by 2, namely $\max \|HITS(\mathcal{G}_1) - HITS(\mathcal{G}_2)\|_1 \leq 1$.

3) SALSA: Similar to HITS, the results of SALSA are naturally bounded between 0 and 1. Thus, we have $\max \|SALSA(\mathcal{G}_1) - SALSA(\mathcal{G}_2)\|_1 \leq 1$.

4) ALS: As has been proved in existing work [51], the sensitivity of ALS is shown in Table III. Note that in this work, we adopt the unbounded case.

C. Sensitivity studies on ALS

In addition to the sensitivity studies using PageRank, HITS and SALSA, we also studied the sampling rate parameter for ALS application by varying the sampling rate from 0%, 10%, 20%, . . . , to 100% using MovieLens-1M dataset while keeping the other parameters as default. Figure 16 shows the results.

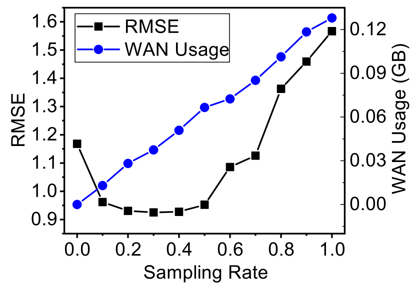


Fig. 16 Sensitivity study on the sampling rate using MovieLens-1M.

The figure clearly shows that with the increase of sampling rate, the WAN usage of the application increases almost linearly while the RMSE first decreases and then increases abruptly. The reason that RMSE first decreases is that with more inter-DC messages being sampled, more useful information participates in graph processing and thus can help to improve the accuracy of processing results. When the sampling rate further increases, the benefit brought by the budget amplification effect of sampling becomes less significant. Thus, the goal of selecting a good sampling rate is to achieve a good balance between the accuracy improvement brought by privacy budget amplification and the accuracy loss due to dropping inter-DC messages. By default, we set the sampling rate to 30% for ALS.