Probabilistic Inference Based Incremental Graph Index for Similarity Search on Social Networks

Tong Lu[1,2], Zhiwei Qi[1,2(✉)], Kun Yue[1,2], and Liang Duan[1,2]

[1] School of Information Science and Engineering, Yunnan University, Kunming, China
mckinleylu@mail.ynu.edu.cn, {kyue,duanl,maryqizhiwei}@ynu.edu.cn
[2] Yunnan Key Laboratory of Intelligent Systems and Computing, Yunnan University, Kunming, China

Abstract. To find k neighbor users on social networks, the efficient approximate nearest neighbor search (ANNS) is useful. Existing graph index methods have shown attractive performance, but suffer from inaccuracy w.r.t. unindexed queries. To achieve both indexed and unindexed queries for graph-index methods, we propose an incremental graph index based method for ANNS on social networks. First, graph convolutional network based on attention mechanism is adopted to embed the social network into low-dimensional vector space, on which the graph index is constructed efficiently. To add the unindexed queries to the graph index incrementally, we propose Bayesian network (BN) learned from social interactions to represent dependency relations of unindexed queries and their neighbors, and perform probabilistic inferences in BN to infer the closest neighbors of unindexed queries. Extensive experiments show that our proposed method outperforms the state-of-the-art methods on both execution time and precision.

Keywords: Social network · Similarity search · Incremental graph index · Bayesian network · Probabilistic inference

1 Introduction

Millions of people use various social networks to connect with friends and family, and share private information. A key issue in social network is to find out the prospective friends of users so as to extend the users' social cycles. For example, Facebook has a friend finding page that suggests people you may know based on factors like mutual friends, shared workplaces or similar social interactions. Approximate nearest neighbor search (ANNS), as a similarity search method, is more widely utilized to find k neighbors of users by constructing efficient index on large-scale and high-dimensional social networks, striking a better trade-off between accuracy and efficiency [24]. As shown in Fig. 1 (a), ANNS starts with a navigating node and takes a greedy routing strategy to find the query's k

H. Gao et al. (Eds.): CollaborateCom 2023, LNICST 562, pp. 458–473, 2024.
https://doi.org/10.1007/978-3-031-54528-3_25

($k = 3$) neighbors by calculating the similarity based on Euclidean distance among nodes.

Graph-index ANNS [7,8,23] methods show attractive performance for indexed queries (queries in graph index). However, these methods do not support unindexed queries (queries not in graph index) that lead to the curse of inaccurate search results. For example, although indexed query is closer to unindexed query than other indexed nodes, indexed query's neighbors do not include unindexed query in Fig. 1 (a). When unindexed query is added into graph index, it becomes indexed query's neighbor in Fig. 1 (b) that improves the accuracy of search results. Besides, graph-index ANNS cannot be directly fulfilled on social network but feature vectors [18], where the feature vectors, i.e. node embeddings of a social network, are commonly used to construct graph index. To this end, we will address the following problems to update unindexed queries into graph index: 1) how to map social network to obtain node embeddings and construct graph index on node embeddings; 2) how to describe the dependency relations of nodes by making fully use of social interactions; 3) how to add the edges of unindexed queries and their neighbors efficiently and accurately.

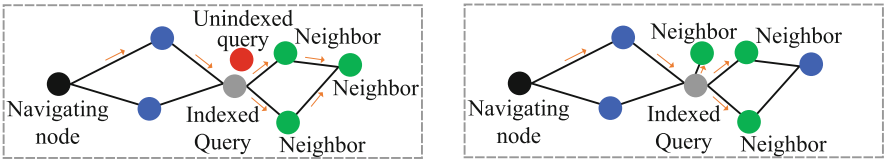

Fig. 1. Search results of the indexed query on graph index vs incremental graph index. (a) the search results (green circles) of indexed query do not include the unindexed query. (b) the search results of indexed query include the unindexed query. (Color figure online)

To address the aforementioned problems and improve the performance of ANNS, our **P**robabilistic **I**nference based **I**ncremental **G**raph **I**ndex (PI-IGI) for similarity search on Social Networks is proposed. The graph index could be constructed efficiently on node embeddings rather than the social network itself. To map the social network into vector space while preserving network structure, an two-layer graph convolutional network (GCN) is proposed to generate node embeddings by adding a masked graph attention mechanism. Based on the similarity calculation of node embeddings, the initial graph index, also named KNNG, could be easily constructed. To keep the neighbors distributed around the nodes of KNNG, we prune the edges by constraining the angles of edges. A depth-first-search (DFS) tree is also built to keep the connectivity of graph index.

As shown in Fig. 1 (b), the increment graph index could improve the accuracy of search results. To update the unindexed queries into graph index, a candidate selection method is proposed to generate enough candidate neighbors to connect with unindexed queries. The Bayesian network (BN), including a

directed acyclic graph (DAG) and conditional probability tables (CPTs), is used to qualitatively and quantitatively describe the dependency relations of random variables [16,22]. To describe the dependency relations between unindexed queries and their candidate neighbors based on social interactions, we propose the concept of query neighbor based Bayesian network (QNBN) and learn the DAG and CPTs of QNBN by rule-based method. The nodes of QNBN correspond to unindexed queries and their candidate neighbors, and edges correspond to the dependency relations of unindexed queries and their candidate neighbors. On the learned QNBN, we perform approximate inferences to infer the potential edges of unindexed queries and their candidate neighbors to be added into the graph index. Furthermore, a neighbor evaluation function is proposed to estimate which edges are added into the graph index, where the function combines Euclidean distance and posterior probabilities of nodes. Finally, the top k neighbors of each unindexed query could be obtained by performing ANNS on the graph index.

The experimental results demonstrate that our PI-IGI improves the search precision by at least 5% and the search speed by at least 10% compared with the state-of-the-art methods.

The rest of this paper is organized as follows: Sect. 2 shows the related work. Section 3 presents our method PI-IGI. Section 4 provides the experimental results of PI-IGI compared to state-of-the-art methods. We give the conclusion and future work in Sect. 5.

2 Related Work

Non-graph-index ANNS. The non-graph-index methods mainly include tree-based, permutation-based, and hashing-based methods in Fig. 2. Tree-based methods partition data by their dimensions, including KD-tree [3], PM-tree [30], and NV-tree [15]. Hashing-based methods divide the data by hash functions and map the divided data into hyper-surfaces, including locality-sensitive hashing [4], spectral hashing [11] and minwise hashing [17]. Permutation-based methods use codebooks to encode data that could reduce memory consumption, including variance-aware quantization [20] and product quantization [19]. However, non-graph-index methods are much less efficient than graph-index ones when performing ANNS on large-scale social networks, due to traversing more nodes than graph-index methods. Therefore, we need to propose a graph-index ANNS method to improve the search efficiency.

Graph-index ANNS. Graph-index methods in Fig. 2 are more efficient and effective than non-graph-index methods for ANNS [24]. The state-of-art methods [7,8,18,23] improve the performance of ANNS by using different edge-select strategies. HCNNG [18] uses a hierarchical clustering algorithm to build graph index with a balanced tree structure, decreasing the memory consumption of graph index. NSG [8] builds a monotonic graph index to shorten the search path and obtains better performance of ANNS compared to HCNNG. GI-DSNE [23] is constructed by pruning edges on pre-built KNNG [21] and employs depth-first-search (DFS) trees to keep the connectivity of index, but suffers the curse

of search precision of unindexed queries. NSSG [7] constrains the edges between two nodes to adjust the sparsity of index to guarantee the monotonicity of the search path of unindexed queries. However, these graph-index methods ignore the incremental updates of graph index.

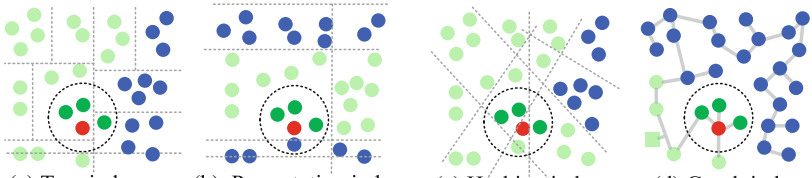

Fig. 2. Different base indexes on the identical dataset. The red circle represents query node, while dark green, light green and blue circles represent the nearest neighbors, checked and unchecked nodes, respectively. (Color figure online)

3 Methodology

In this section, we introduce our proposed method PI-IGI in Fig. 3 that includes graph index construction, QNBN learning, incremental graph index construction, and ANNS on graph index.

3.1 Graph Index Construction of Social Network

The formal definitions of social network, social network embedding and graph index are presented at first.

Definition 1. A social network is denoted as $G = (V, E)$, where $V = \{v_i\}_{i=1}^{n}$ is a set of nodes representing users, $E = \{e_{ij}\}_{i,j=1}^{n}$ is a collection of edges. $\mathbf{A} = \{a_{ij}\}_{i,j=1}^{n}$ is an adjacency matrix of G, where $a_{ij} = 1$ if and only if there is an edge e_{ij} between v_i and v_j, vice versa, $a_{ij} = 0$.

Definition 2. The embedding $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^{n}$ of G is a mapping f: v_i to $\mathbf{y}_i \in \mathbb{R}^d$, where $d << n$ and the function f preserves the structure of G.

Definition 3. A index $G^I(V^I, E^I, \delta)$ is built on $\mathbf{Y}$ by graph-index methods, where $V^I = \{v_i^I\}_{i=1}^{n}$ is a set of nodes, $E^I = \{e_{ij}^I\}_{i,j=1}^{n}$ is a set of edges, and $\delta_{i,j} \in \delta$ is the similarity function based on the Euclidean distance (l_2 norm) between v_i^I and v_j^I in Eq. 1.

$$\delta_{i,j} = \delta(v_i^I, v_j^I) = \left(\sum_{l=0}^{d-1} (v_{il}^I - v_{jl}^I)^2 \right)^{\frac{1}{2}} \tag{1}$$

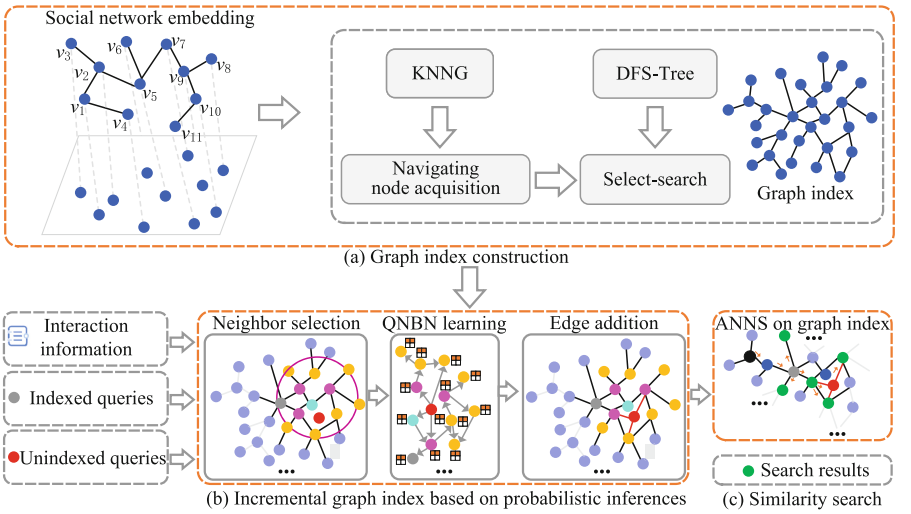

Fig. 3. Overview of our proposed PI-IGI. (a) the graph index is constructed on social network embedding. (b) the QNBN is learned on selected candidate neighbors of unindexed queries (red circles) and edges of incremental graph index are added by performing probabilistic inferences in QNBN. The cyan, pink and yellow circles represent the nearest candidate neighbors of unindexed queries, one-hop and two-hop neighbors of cyan circles, respectively. (c) the search results (green circles) of unindexed queries and indexed queries is obtained by ANNS on graph index. (Color figure online)

Definition 4. Social interactions characterizing a set of node pairs $\Omega = \{< v_i, v_j > | v_i, v_j \in V, i \neq j\}$ is denoted as an adjacency matrix $\mathbf{M}$, where $< v_i, v_j >$ denotes an social interaction between v_i and v_j.

To construct graph index for ANNS on low-dimensional vector space, graph convolutional network (GCN) based on attention mechanism is proposed to map social network to generate node embeddings. In order to preserve the structure of social network effectively, the social network is transformed to a node degree matrix $\mathbf{D}$ and adjacency matrix $\mathbf{A}$ as inputs of two-layer GCN. In real social network, the neighbors of a node should be assigned to different weights because of different relationship strengths among nodes. To ensure important neighbors obtain large weights, the masked graph attention mechanism [29] concatenated to GCN aims to allocate weight to different neighbors. The node embedding $\mathbf{y}_i$ is generated by averaging aggregated features from each attention head as follows

$$\mathbf{y}_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \beta_{ij} \mathbf{W} \cdot \mathbf{z}_j\right) \tag{2}$$

where $\mathbf{Z} = \mathrm{GCN}(\mathbf{D}, \mathbf{A})$, $\mathbf{z}_j \in \mathbf{Z}$ denotes the initial node embedding, $\mathbf{W} = \mathrm{GCN}(\mathbf{M}, \mathbf{A})$, $\mathbf{W}$ denotes weight matrix, $\mathcal{N}_i$ denotes the neighbor set of $\mathbf{z}_i$, and β_{ij} denotes attention coefficient between $\mathbf{z}_i$ and $\mathbf{z}_j$.

To speed up the construction of graph index, we construct a graph index based on KNNG. Considering that the KNNG has plenty of edges that give rise to detours in the search path and memory consumption. In order to improve the efficiency of ANNS and guarantee the monotonicity of search path, we prune the edges of KNNG, cut long edges and control the angles among edges from 0° to 60° by ANNS starting from navigating nodes that is the centroid of data. To ensure the connectivity of graph index, DFS tree is spanned to merge possible connected indexes.

3.2 QNBN Learning on Social Interactions

To add unindexed queries into graph index efficiently, the enough candidate neighbors of unindexed queries are first obtained from graph index. Then, QNBN is learned from frequent itemset based on social interactions to describe the dependency relations of unindexed queries and their candidate neighbors. The frequent itemset and QNBN are defined as

Definition 5. The frequent itemset F consists of items in social interactions and satisfies the following condition

$$Support(F) = P(F; \Omega) \geq min_{support} \tag{3}$$

where Ω is a set of social interactions, $P(F; \Omega)$ denotes the frequency of F occurred in Ω, and $min_{support}$ denotes the threshold of minimum support.

Definition 6. $B = (\mathcal{G}, \theta)$ represents a QNBN with a directed acyclic graph $\mathcal{G}$ and conditional probability parameters θ, where

- $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a DAG, $\mathcal{V} = \{X_i\}_{i=1}^{n_\varepsilon}$ is a collection of nodes of DAG, $\mathcal{E} = \{r_{ij}\}_{i,j=1}^{n_\varepsilon}(i \neq j)$ is a set of edges of DAG, X_i represents a node or a conjuction of nodes and X_i takes the values of 0 or 1.
- θ is a collection of probability parameters of nodes $\mathcal{V}$.

Candidate Neighbor Selection of Unindexed Queries. To generate the candidate neighbors of unindexed query v_q, we propose a candidate neighbor selection method based on NN-Descent [5]. The node v_p^I having the minimum Euclidean distance with v_q is first obtained by traversing the nodes in graph index and the one-hop and two-hop neighbors (orange nodes and yellow nodes in Fig. 3 (b)) of v_p^I are considered as candidate neighbors of v_q, that is, the neighbors' neighbors of v_p^I are also its neighbors, defined as

$$V_s^I(v_p^I) = \{v_i^I | v_i^I \in G^I \wedge \delta(v_p^I, v_i^I) <= \max(\delta(v_p^I, v_i^I))\},\ v_p^I \in V_s^I(v_p^I). \tag{4}$$

where V_s^I denotes a set of candidate neighbors of v_p^I, $\delta(v_p^I, v_i^I)$ represents the Euclidean distance between of pair node, $\max(\delta(v_p^I, v_i^I))$ denotes the maximum Euclidean distance of node v_p^I and its one-hop neighbor.

The number of candidate neighbors of v_q is further reduced by clustering its candidate neighbors to improve the efficiency of the update of graph index. To cluster candidate neighbors efficiently, k-means++ algorithm is adopted due to its better convergence and possessing a more intelligent initialization strategy for centroid placement [6].

Social interactions imply the dependency relations of unindexed queries and their neighbors, which are used to obtain query neighbor rules (QNRs) to learn the structure of QNBN. The QNBN learning includes structure learning and parameter learning, where the QNR needs to satisfy two properties:

1) The more frequently between two nodes of G^I interact, the stronger the dependency relations between nodes are, i.e., the more frequent interactions $< v_i^I, v_j^I >$ between v_i^I and v_j^I are, the more possibly QNR $v_i^I \rightarrow v_j^I$ between them is occurred.
2) The shorter hop between two nodes of G^I are, the more possibly QNR $v_i^I \rightarrow v_j^I$ between them is occurred, i.e., if there exists $v_i^I \rightarrow v_j^I$ and $v_i^I \rightarrow v_k^I \rightarrow v_j^I$, the more possibly QNR $v_i^I \rightarrow v_j^I$ exists.

Structure Learning. To generate QNRs to learn the structure of QNBN, a set of frequent itemsets F is first obtained by Eq. 3, where each QNR $l_s \rightarrow (l - l_s)$ extracted from F satisfies $\frac{support(l)}{support(l_s)} \geq \alpha$, and α is the threshold of the minimum confidence of QNR. The confidence of a QNR $v_i^I \rightarrow v_j^I$ could be formulated as

$$confidence(v_i^I \rightarrow v_j^I) = P(v_j^I | v_i^I) = \frac{support(v_i^I \cup v_j^I)}{support(v_i^I)} \tag{5}$$

where $support(v_i^I \cup v_j^I)$ is number of social interactions of v_i^I or v_j^I, $support(v_i^I)$ is the number of social interactions of v_i^I.

Further, each QNR between unindexed query v_q and its candidate neighbors is viewed as

$$v_1^I \wedge v_2^I \wedge \ldots \wedge v_n^I \rightarrow v_q \tag{6}$$

On the basis of logical implication, QNR $v_1^I \wedge v_2^I \wedge \ldots \wedge v_n^I \rightarrow v_q$ is equivalently transformed into Horn clause $\overline{v_1^I} \vee \overline{v_2^I} \vee \ldots \overline{v_n}^I \vee \overline{v_1^I \wedge v_2^I \wedge \ldots \wedge v_n^I} \vee v_q$. Once all Horn clauses are transformed into DAGs, we merge DAGs to generate the ultimate DAG of QNBN.

Parameter Learning. To calculate the dependency relations between unindexed queries and their candidate neighbors quantitatively, we learn the CPTs of each node in DAG. If the nodes have no parents, their prior probabilities are evaluated by normalizing the number of frequent nodes in social interactions. If the nodes have parents, we employ logical constraints represented by Boolean expressions to evaluate their conditional probability

$$P(X_i = k^i | \pi(X_i)) = (b_1, ..., b_z) = f_{X_i \pi(X_i)} = \begin{cases} 1, & k^i = b_1 \vee ... \vee b_z \\ 0, & otherwise \end{cases} \tag{7}$$

where $\pi(X_i)$ denotes the parents of X_i, $k^i, b_j \in \{0,1\}(1 \leq i, j \leq z)$.

3.3 Probabilistic Inferences for Incremental Graph Index

To connect unindexed query with its candidate neighbors in the graph index efficiently, approximate inference based on random sampling [22] is employed to calculate posterior probability to quantitatively describe their dependence relations. The posterior probability $P(X_{ik^i}|X_{jk^j})$ is formulated as

$$P(X_{ik^i}|X_{jk^j}) \approx \frac{\#(X_i = k^i)}{n_s} \tag{8}$$

where $k^i \in (0,1)$ is the set of possible values of X_i and n_s represents the sample size. X_{ik^i} corresponds to v_i^I of G^I or unindexed query, if a node v_i^I is in the graph index, then the value k^i of X_i is 1, i.e., $X_i = 1$.

To add the unindexed query into graph index accurately, a nearest neighbor evaluation is proposed to measure the similarity of unindexed query and its nearest neighbors of graph index. The nearest neighbor evaluation $\Delta'(v_i^I, v_q)$ includes evaluations of structural similarity and conditional probability of unindexed query and its nearest neighbors, formulated as

$$\Delta'(v_i^I, v_q) = \delta(v_i^I, v_q) + W \tag{9}$$

where W is the penalty term defined as

$$W = -\frac{P(X_q = 1|X_{v_i^I} = 1)\delta(v_i^I, v_q)}{n_{v_i^I v_q}} \tag{10}$$

where the numerator part of the penalty term consists of posterior probability and Euclidean distance among nodes, $n_{v_i^I v_q}$ denotes the number of social interactions of v_i^I or v_q, and $P(X_q = 1|X_{v_i^I} = 1)$ denotes the probability of unindexed query $X_{v_q} = 1$ given $X_{v_i^I} = 1$.

According to Eq. 8, $P(X_q = 1|X_{v_i^I} = 1)$ could be rewritten as $\frac{\#(X_q = k^q)}{n_s}$ and k^q is set to 1, the neighbor evaluation function is formulated as

$$\Delta'(v_i^I, v_q) = \delta(v_i^I, v_q) + W \approx \delta(v_i^I, v_q) - \frac{\#(X_q = 1)\delta(v_i^I, v_q)}{n_s \cdot n_{v_i^I v_q}} \tag{11}$$

The neighbor relationships between unindexed queries and their candidate neighbors are calculated by Eq. 11. The top k neighbors of each unindexed query could be obtained via the calculation results. Thus, the incremental graph index is built by adding edges between unindexed queries and their top k neighbors.

3.4 ANNS on Graph Index

For indexed queries, their k nearest neighbors are obtained by performing ANNS on the graph index. For unindexed queries, we incrementally connect them into the graph index and perform ANNS to find their k nearest neighbors. The ideas of PI-IGI are presented in Algorithm 1.

Algorithm 1. ANNS on incremental graph index(G^I,v_p^I,v_q,H)

Input: Graph index $G^I = (V^I, E^I, \delta)$, navigating node v_p^I, query node v_q, social interactions set H;
Output: k nearest neighbors of v_q;

```
traverse the nodes to judge whether v_q is an unindexed query or not.
if v_q is an unindexed query then
    v_i^I ← the node of index has nearest neighborhood relationship with v_q.
    obtain candidate neighbor set C_s^I of v_q.
    learn QNBN on C_s^I.
    perform probability inferences on QNBN to obtain the nodes set S having dependency relations with v_q.
    for all each node v_i^I in S do
        if Δ'(v_q, v_i^I) < Δ'_max then
            add edge e_qi into G^I.
            Δ'_max ← Δ'(v_q, v_i^I).
            if the number of v_i^I's neighbors > l then
                remove the node that has the minimum similarity with v_i^I.
            end if
        end if
    end for
end if
return k nearest neighbors by performing ANNS over graph index.
```

The time complexity of steps $3 \sim 4$, steps $5 \sim 6$, steps $7 \sim 16$ and step 17 is $O(n) + O(k) + O(k^3d + T_1k^3d)$, $O(|\sum|) + O(|S|)$, $O(|S|)$ and $O(\log n)$, respectively. T_1 is the number of iterations of k-means++ algorithm, $|\sum|$ is the number of QNRs. The complexity of Algorithm 1 is $O(|n|)$.

4 Experiments

In this section, our approach PI-IGI is compared with existing non-graph-index and graph-index methods. Two sets of experiments are conducted on social networks: 1) the efficiency of social network embedding and graph index updating; 2) the effectiveness and efficiency of ANNS on PI-IGI. Each experiment was repeated for five times, we exhibit the average evaluations of social network embedding, graph index updating and ANNS.

4.1 Experimental Setting

Datasets. Real-world social networks such as Facebook[1], soc-Reddithyperlink (Soc)[2], Twitter[3] and Weibo[4] are employed and all network datasets contain social interactions. Facebook describes communication among people over time. Soc is generated by extracting available Reddit data. Twitter and Weibo contain lots of social interactions among users, such as comments and retweeted posts. Table 1 provides more details, where $\#(Interaction)$ represents the number of social interactions, $\#(Edge)$ represents the number of edges and $\#(Query)$ represents the number of unindexed queries.

Table 1. Details of social networks.

Dataset	$\#(Interaction)$	$\#(Edge)$	$\#(Query)$
FaceBook	33,720	7089	10
Soc	2,712,512	858,490	700
Twitter	4,012,396	1,003,564	1000
WeiBo	4,178,974	1,054,196	1000

Comparison Methods. Our method PI-IGI was compiled by using C++14 to compare with existing graph-index methods and non-graph-index methods. GI-DSNE has an extra step to embed social network into low-dimensional vector space, while other comparative methods construct indexes on low-dimensional vectors directly. We performed the experiments on a machine running the Ubuntu operating system, equipped with a ten-core Intel Core i9-10900X CPU operating at 3.7 GHz and 128GB of main memory.

- **GI-DSNE** [23] embeds social network into low dimensional vector space, constructs the graph index on node embeddings, and cuts redundant edges of graph index to improve the construction time of graph and precision of ANNS.
- **NSSG**[5] [7] prunes the edges via constraining angle between two edges and guarantees search path being monotonic.
- **NSG**[6] [8] constructs a monotonic graph index based on MRNG and regards the global centroid of the graph index as the navigating node.
- **FANNG**[7] [10] selects more discretely distributed neighbors from more than k nearest vertices based on RNG.

[1] https://nrvis.com/download/data/dynamic/.
[2] https://snap.stanford.edu/data/soc-RedditHyperlinks.html.
[3] https://www.aminer.cn/data-sna#Twitter-Dynamic-Net.
[4] https://aminer.org/Influencelocality.
[5] https://github.com/ZJULearning/SSG.
[6] https://github.com/ZJULearning/nsg.
[7] https://github.com/ZJULearning/efannagraph.

- **HCNNG**[8] [18] constructs graph index based on MST and its navigating node is obtained on building KD-tree.
- **FAISS**[9] [12] is a non-graph index based permutation, which has both CPU and GPU versions, with the CPU version being employed.
- **FALCONN**[10] [1] based on locality-sensitive hashing (LSH) has excellent search performance on low-dimensional datasets.
- **ANNOY**[11] [2] is a tree-based index built by constructing an elegant binary space partition (BSP) tree.

Evaluation Metrics. We used t to evaluate the efficiency of network embedding. The parameters d, α and m were varied to evaluate the update time of PI-IGI. The search performance of PI-IGI was evaluated by calculating the queries per second (QPS) and the search precision of ANNS. Precision is defined as $precision = \frac{|R \cap G|}{|G|}$, where R denotes query result set and G denotes the real result set.

4.2 Experimental Results

Efficiency of Social Network Embedding and Graph Index Updating. We recorded the execution time of social network embedding and graph index updating on different datasets by varying the dimension d of embeddings and the confidence threshold α of QNRs selection, respectively, to evaluate their efficiency.

Exp-1.1 Impacts of d on the execution time of social network embedding. PI-IGI is compared with GI-DSNE because of these methods including extra embedding step. To estimate the execution time of embedding on different datasets in Table 1, d was varied from 128 to 512. Table 2 shows the results of embedding execution time. For each dataset, we found that the larger the dimension is, the longer the time of embedding is, such as $d = 512$ taking about 0.48 times more embedding time than that of $d = 128$ on Facebook, since the more the dimension of embeddings, the longer the time of embedding computation. Besides, the larger the dataset is , the more time the embedding takes, such as Soc taking about 150 times more embedding time than that of FaceBook on the same dimension $d = 128$. Moreover, our embedding method is more efficient than GI-DSNE under the same condition. This is because the masked graph attention mechanism incorporates linear transformations of features to map the original high-dimensional features to a lower-dimensional space, learning to reduce the computational complexity of embedding.

To demonstrate the efficiency of graph index updating on all datasets in Table 1, we evaluated the impacts of different confidences α of QNR selection and various embedding dimensions d of datasets on the update time of PI-IGI.

[8] https://github.com/jalvarm/hcnng.
[9] https://github.com/facebookresearch/faiss.
[10] https://github.com/FALCONN-LIB/FALCONN.
[11] https://github.com/spotify/annoy.

Table 2. Impacts of data size and date dimension on the execution time of graph embedding methods.

Dataset	Embedding time(s)						
	d=128	d=192	d=256	d=320	d=384	d=448	d=512
Facebook(GI-DSNE)	17.0637	33.63	54.25	68.94	92.31	103.23	115.48
Facebook(PI-IGI)	**2.49**	**2.66**	**2.79**	**3.04**	**3.24**	**3.58**	**3.71**
Soc(GI-DSNE)	9264.246	10581.97	15962.47	27994.11	33427.21	40892.55	52911.96
Soc(PI-IGI)	**347.95**	**369.60**	**403.90**	**429.00**	**462.77**	**502.63**	**549.70**
Twitter(GI-DSNE)	13278.47	20742.23	26117.31	37945.17	50720.07	59429.64	72145.37
Twitter(PI-IGI)	**1180.26**	**1297.54**	**1387.17**	**1483.39**	**1561.77**	**1683.47**	**1767.71**
WeiBo(GI-DSNE)	13496.97	22717.90	29047.47	38125.68	51990.84	61294.23	74001.23
WeiBo(PI-IGI)	**1206.19**	**1305.00**	**1415.49**	**1492.68**	**1574.51**	**1694.82**	**1789.40**

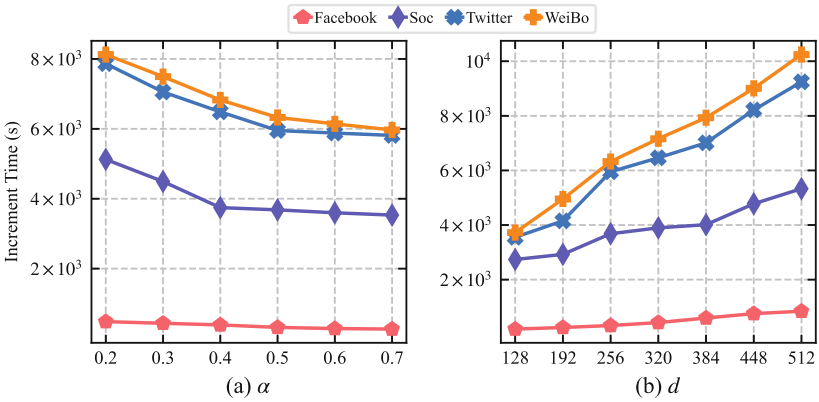

Fig. 4. Impacts of α and d on the execution time of graph index updating.

Exp-1.2 Impacts of confidence α of QNR selection on the update time of graph index. The confidence α of QNR selection affects the update time of graph index. We find in Fig. 4 (a) that the time will be longer when α becomes smaller. This implies that the smaller the confidence threshold α is, the more QNRs are utilized to learn QNBN.

Exp-1.3 Impacts of embedding dimension d on the update time of graph index. To discover the impact of different dimensions of datasets on the update time of graph index, d was varied from 128 to 512. The results show that the larger the dimension is, the more time the update of graph index costs in Fig. 4 (b). This is because the higher the dimension of embeddings is, the longer time the PI-IGI takes to train GCN and masked graph attention mechanism.

Effectiveness and Efficiency of ANNS. To demonstrate the effectiveness and efficiency of ANNS on PI-IGI, we adopted QPS vs search precision to compare PI-IGI with five graph-index methods and three non-graph-index methods.

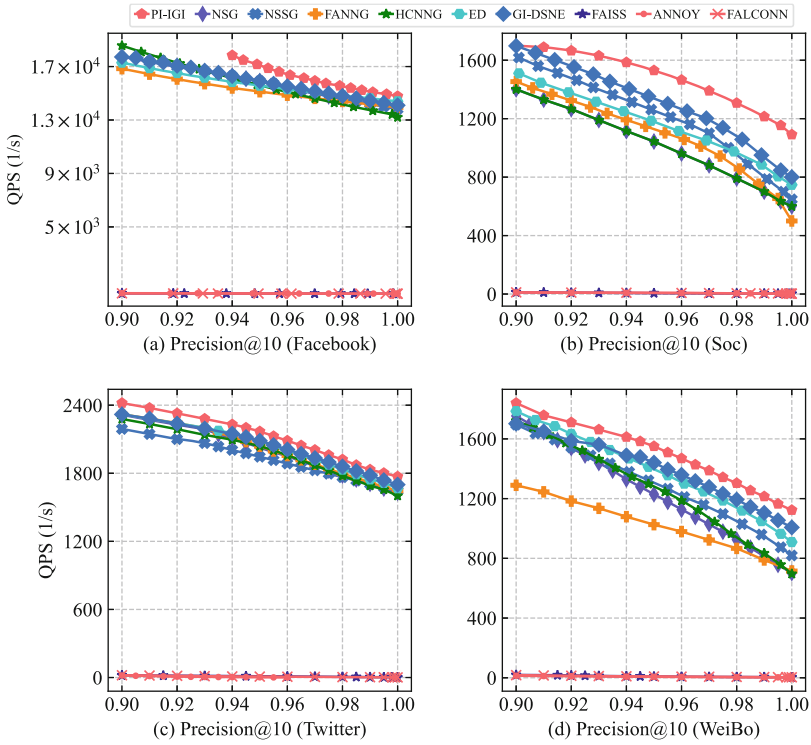

Fig. 5. QPS Vs Precision.

Exp-2.1 Queries Per Second vs Search Precision. For PI-IGI, NSG and NSSG, the quality and the size of graph indexes were set to 100 and 50, respectively. Meanwhile, the angles between two edges were set to 50 and the maximum candidate pool size was set to 100. For all compared methods, the dimension of datasets, the quality of search results and the number of neighbors were set to 256, 20 and 15, respectively. The QPS vs search precision of each indexing method is reported in Fig. 5. The results are as follows:

1) The search efficiency of PI-IGI is higher than that of other methods. For instance, the QPS of PI-IGI is higher by about 20% than that of GI-DSNE on Twitter when the precision attains 0.90. This is because that PI-IGI is an incremental graph index, the search path of unindexed queries' k neighbors becomes shorter.
2) Given same QPS and dataset, the precision of PI-IGI is higher than that of other methods. For example, the precision of PI-IGI is about 5% higher than that of GI-DSNE on Facebook by fixing the QPS as 500, since unindexed queries could be found in search results.
3) It is not difficult to find that graph-index methods achieve much better ANNS performance than non-graph-index methods.

Exp-2.2 Ablation Experiment. To verify the effectiveness and efficiency of PI-IGI, we conducted incremental graph index method based on Euclidean distance (ED) to compare with PI-IGI. As shown in Fig. 5, the QPS of PI-IGI is higher than that of ED at the same precision, such that the QPS of PI-IGI is approximately 10% higher than that of ED on WeiBo when the search precision attains 0.98. This implies that PI-IGI takes shorter path to search neighbors of unindexed queries. Moreover, the search precision of PI-IGI is higher than that of ED at the same QPS, such that the precision of PI-IGI is approximately 3% higher than that of ED on WeiBo by fixing the QPS as 1200. This is because that PI-IGI obtains more accurate neighbors by calculating the dependency relations of unindexed queries and their candidate neighbors.

5 Conclusion and Future Work

We present a probabilistic inference based incremental graph index for ANNS, which improves search precision by at least 5% and search efficiency by 10% compared to the state-of-the-art methods. The improvements are due to the addition of unindexed queries into the graph index. Besides, we propose QNBN to describe the dependency relations between unindexed queries and their candidate neighbors and perform approximate inference in QNBN to infer the potential edges of graph index. Theoretical analysis and extensive experiments manifest the advantages of our method. In the future, we will propose a novel method based on Bayesian neural network to obtain accurate dependency relations of nodes to improve the precision of ANNS on PI-IGI.

Acknowledgments. This paper was supported by the National Natural Science Foundation of China (62002311), Major Project of Science and Technology of Yunnan Province (202202AD080001), Yunnan Key Laboratory of Intelligent Systems and Computing (202205AG070003), Research Foundation of Educational Department of Yunnan Province (2023J0022).

References

1. Andoni, A., Indyk, P., Laarhoven, T., Razenshteyn, I., Schmidt, L.: Practical and optimal lsh for angular distance. In: Advances in Neural Information Processing Systems, 28 (2015)
2. Bernhardsson, E.: Annoy at github. GitHub. Repéré à https://github.com/spotify/annoy (2015)
3. Bi, W., Ma, J., Zhu, X., Wang, W., Zhang, A.: Cloud service selection based on weighted KD tree nearest neighbor search. Appl. Soft Comput. J. **131**, 109780 (2022)
4. Cheng, D., Huang, J., Zhang, S., Wu, Q.: A robust method based on locality sensitive hashing for k-nearest neighbors searching. Wireless Networks, pp. 1–14 (2022)
5. Dong, W., Moses, C., Li, K.: Efficient k-nearest neighbor graph construction for generic similarity measures. In: Proceedings of the 20th international conference on World wide web, pp. 577–586 (2011)

6. Ezugwu, A.E., et al.: A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. Eng. Appl. Artif. Intell. **110**, 104743 (2022)
7. Fu, C., Wang, C., Cai, D.: High dimensional similarity search with satellite system graph: Efficiency, scalability, and unindexed query compatibility. IEEE Trans. Pattern Anal. Mach. Intell. **44**(8), 4139–4150 (2021)
8. Fu, C., Xiang, C., Wang, C., Cai, D.: Fast approximate nearest neighbor search with the navigating spreading-out graph. Proc. VLDB Endow. **12**(5) (2019)
9. Gorunescu, F.: Data Mining: Concepts, models and techniques. Springer Science, San Francisco (2011)
10. Harwood, B., Drummond, T.: Fanng: Fast approximate nearest neighbour graphs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5713–5722 (2016)
11. Hu, D., Nie, F., Li, X.: Discrete spectral hashing for efficient similarity retrieval. IEEE Trans. Image Process. **28**(3), 1080–1091 (2018)
12. Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with gpus. IEEE Trans. on Big Data **7**(3), 535–547 (2019)
13. Kosuge, A., Yamamoto, K., Akamine, Y., Oshima, T.: An soc-fpga-based iterative-closest-point accelerator enabling faster picking robots. IEEE Trans. Industr. Electron. **68**(4), 3567–3576 (2020)
14. Lejsek, H., Amsaleg, L.: Nv-tree: an efficient disk-based index for approximate search in very large high-dimensional collections. IEEE Trans. Pattern Anal. Mach. Intell. **31**(5), 869–883 (2008)
15. Lejsek, H., Amsaleg, L.: Nv-tree: nearest neighbors at the billion scale. In: Proceedings of the 1st ACM International Conference on Multimedia Retrieval, pp. 1–8 (2011)
16. Li, J., Yue, K., Li, J., Duan, L.: A probabilistic inference based approach for querying associative entities in knowledge graph. In: Proceedings of the Web and Big Data: 5th International Joint Conference, pp. 75–89 (2021)
17. Li, P., Shrivastava, A., Moore, J., König, A.: Hashing algorithms for large-scale learning. In: Advances in Neural Information Processing Systems 24 (2011)
18. Munoz, J.V., Gonçalves, M.A., Dias, Z., Torres, R.d.S.: Hierarchical clustering-based graphs for large scale approximate nearest neighbor search. Pattern Recognition **96**, 106970 (2019)
19. Pan, Z., Wang, L., Wang, Y., Liu, Y.: Product quantization with dual codebooks for approximate nearest neighbor search. Neurocomputing **401**, 59–68 (2020)
20. Paparrizos, J., Edian, I., Liu, C., Elmore, A.J., Franklin, M.J.: Fast adaptive similarity search through variance-aware quantization. In: 2022 IEEE 38th International Conference on Data Engineering, pp. 2969–2983 (2022)
21. Paredes, R., Chávez, E.: Using the k-nearest neighbor graph for proximity searching in metric spaces. In: Proceedings of the String Processing and Information Retrieval: 12th International Conference, pp. 127–138 (2005)
22. Qi, Z., Yue, K., Duan, L., Hu, K., Liang, Z.: Dynamic embeddings for efficient parameter learning of Bayesian network with multiple latent variables. Inf. Sci. **590**, 198–216 (2022)
23. Qi, Z., Yue, K., Duan, L., Liang, Z.: Similarity search with graph index on directed social network embedding. In: Web Engineering: 22nd International Conference, pp. 82–97 (2022)
24. Wang, M., Xu, X., Yue, Q., Wang, Y.: A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search. Proc. VLDB Endowment **14**(11), 1964–1978 (2021)

25. Welling, M., Kingma, D.P.: Auto-encoding variational bayes. In: ICLR (2014)
26. Xu, X., Wang, M., Wang, Y., Ma, D.: Two-stage routing with optimized guided search and greedy algorithm on proximity graph. Knowl.-Based Syst. **229**, 107305 (2021)
27. Yap, G.E., Tan, A.H., Pang, H.H.: Explaining inferences in bayesian networks. Appl. Intell. **29**, 263–278 (2008)
28. Yu, S., Sun, Y., Guo, Z.: Graph regularized unsupervised deep hashing for large scale image retrieval. In: 2020 5th IEEE International Conference on Big Data Analytics, pp. 292–297 (2020)
29. Zhang, T., Liu, B., Niu, D., Lai, K., Xu, Y.: Multiresolution graph attention networks for relevance matching. In: Proceedings of the 27th ACM international conference on information and knowledge management, pp. 933–942 (2018)
30. Zheng, B., Zhao, X., Weng, L., Nguyen, Q.V.H., Liu, H., Jensen, C.S.: Pm-lsh: a fast and accurate in-memory framework for high-dimensional approximate nn and closest pair search. VLDB J. **31**(6), 1339–1363 (2022)