Hindawi
Discrete Dynamics in Nature and Society
Volume 2024, Article ID 9897179, 1 page
https://doi.org/10.1155/2024/9897179

Retraction

Retracted: Personalized Influential Community Search in Large Networks: A K-ECC-Based Model

Discrete Dynamics in Nature and Society

Received 23 January 2024; Accepted 23 January 2024; Published 24 January 2024

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope

(2) Discrepancies in the description of the research reported

(3) Discrepancies between the availability of data and the research described

(4) Inappropriate citations

(5) Incoherent, meaningless and/or irrelevant content included in the article

(6) Manipulated or compromised peer review

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] S. Meng, H. Yang, X. Liu, Z. Chen, J. Xuan, and Y. Wu, "Personalized Influential Community Search in Large Networks: A K-ECC-Based Model," *Discrete Dynamics in Nature and Society*, vol. 2021, Article ID 5363946, 10 pages, 2021.

Hindawi
Discrete Dynamics in Nature and Society
Volume 2021, Article ID 5363946, 10 pages
https://doi.org/10.1155/2021/5363946

Research Article

Personalized Influential Community Search in Large Networks: A K-ECC-Based Model

Shi Meng, Hao Yang, Xijuan Liu, Zhenyue Chen, Jingwen Xuan, and Yanping Wu

School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou, China

Correspondence should be addressed to Xijuan Liu; liuxijuan@zjgsu.edu.cn

Received 11 October 2021; Accepted 9 November 2021; Published 29 November 2021

Academic Editor: Gengxin Sun

Graphs have been widely used to model the complex relationships among entities. Community search is a fundamental problem in graph analysis. It aims to identify cohesive subgraphs or communities that contain the given query vertices. In social networks, a user is usually associated with a weight denoting its influence. Recently, some research is conducted to detect influential communities. However, there is a lack of research that can support personalized requirement. In this study, we propose a novel problem, named personalized influential k-ECC (PIKE) search, which leverages the k-ECC model to measure the cohesiveness of subgraphs and tries to find the influential community for a set of query vertices. To solve the problem, a baseline method is first proposed. To scale for large networks, a dichotomy-based algorithm is developed. To further speed up the computation and meet the online requirement, we develop an index-based algorithm. Finally, extensive experiments are conducted on 6 real-world social networks to evaluate the performance of proposed techniques. Compared with the baseline method, the index-based approach can achieve up to 7 orders of magnitude speedup.

1. Introduction

With the proliferation of applications, graphs are widely used to represent entities and their relationships in real-life network data, e.g., social networks, collaboration networks, and communication networks [1–5]. Connected subgraph (community), existing as a functional module in different graphs, has been extensively explored to analyze graphs recently [6]. Community search and community detection are two fundamental problems in graph analysis. Community search aims to identify important communities (i.e., cohesive subgraphs) that contain the query vertices [7], while community detection aims to find all or top-k communities that satisfy the cohesiveness constraint [8, 9]. In this study, we focus on the problem of community search, which is an important tool for personalized applications, such as friend recommendation and product promotion [7, 10, 11].

In social networks, a user is usually associated with a weight, denoting its influence in the network. Recently, the influential community detection problem has attracted great attention (e.g., Ref. [12–15]). Influential community detection aims to find the communities that are not only cohesive but also have large influence value. The influence value of a community is the minimum weight of all the vertices in the community [12, 14]. However, the personalized requirement is ignored by existing research. To meet the requirement, in this study, we propose a new problem, named personalized influential k-edge-connected component (PIKE) search, to find personalized influential communities in social networks. We use the k-edge-connected component (k-ECC) model to measure the cohesiveness of a subgraph, which remains connected after removing any k-1 edges in the graph [16–18]. Given a graph G and a set of query vertices Q, PIKE is the subgraph with the largest influence value that (*i*) contains all the vertices in Q (i.e., personalized), (*ii*) satisfies the k-ECC constraint (i.e., highly connected), and (*iii*) is maximal (i.e., there is no supergraph of it that can meet the constraints in *(i)* and *(ii)*). Note that, in our previous work [19], a k-ECC-based community search model is also proposed. However, it only focuses on the community with maximum k instead of any given k as in this study, while in this study, we can support both scenarios.

Example 1. As shown in Figure 1, it is a small network with 14 vertices. For the simplicity, the number in each vertex denotes both its vertex id and weight. Given a query vertex set $Q = \{v_7, v_8, v_9\}$ and $k = 3$, the vertices in the dotted line are the corresponding result.

1.1. Applications. In the literature, the study of PIKE search problem can find many applications. We list some examples as follows:

(i) Personalized product recommendation: in many social network platforms, such as Facebook, the weight of each user can represent its ability for information promotion in social networks, i.e., viral marketing. The platforms often provide product recommendations for users based on their relationships with others. Given a set of users who are already interested in certain products, other users who are highly connected with them may buy the same products. This is because the highly connected friends may belong to the same social cluster and share similar interests [20]. Besides, the influential users can make the product sales greatly increase. Hence, finding such groups of users in the platforms is helpful for recommendation system, which can be solved by investigating the PIKE of Q (i.e., the most highly connected component containing Q).

(ii) Collaboration team assembling: assembling a collaboration team for a specific project is essential in different scenarios. In a collaboration network, such as DBLP, the weight of vertices can be the influence or impact of the users. The researchers who are highly connected in a collaboration group are good candidates to be invited into the team [21]. Besides, the researchers who have high influence are also easy to get invitations to join groups because they can increase the impact of the research group. Such a team can be obtained by computing PIKE with the set of key researchers or initiators as the query Q. Also, the k-edge-connected component model indicates how strong they are connected.

(iii) Fraud group detection: in e-commerce platforms, such as Amazon, each customer is associated with a weight representing the number of times for its purchases or certain actions. There exist fraudulent users who give fake "like"s to products on platforms in order to promote the product [22]. These fraudulent users often form a closely connected group. Given a set of suspicious customers as the query set, our personalized influential k-ECC model can help us to find the most suspicious fraudster group, which can be further investigated by platforms.

1.2. Challenges. The challenges of the problem are twofold. Firstly, social networks are usually large. Therefore, it requires an algorithm that should have good scalability. Secondly, in real applications, there may be plenty of queries issued. It is necessary that the developed techniques can meet the online requirement.

1.3. Our Solution. To address these challenges, we first propose a baseline algorithm. Due to the definition of PIKE that is the subgraph with the largest influence value, we iteratively remove the vertex with the smallest influence value and maintain the k-ECC model containing the query vertex set. Considering that removing one vertex in each step leads to enormous iterations, especially for the large graph, a dichotomy-based algorithm is proposed. It removes half of the candidate vertices at each iteration and then checks the existence of k-ECC containing the query set, which can efficiently escape the redundant computations and obtain the result. Based on the deletion procedure of baseline, an index-based algorithm is further developed to meet the online requirement. Generally, we keep the order of deleting vertices and construct a tree index for each k. Given the set of query vertices and k, we can first retrieve the tree index by k and then locate the corresponding result efficiently.

1.4. Contributions. To the best of our knowledge, we are the first to investigate the personalized influential k-ECC search problem. The contributions of this study are summarized as follows:

(i) We formally define the personalized influential k-ECC search problem

(ii) Two algorithms, i.e., a baseline algorithm and a dichotomy-based algorithm, are first proposed to address the problem

(iii) To further accelerate the computation and meet the online requirement, an index-based algorithm is proposed

(iv) Experiments over 6 real-world social networks are conducted to show the superiority of proposed techniques

1.4.1. Roadmap. We organize the rest of this study as follows. We first introduce the problem investigated in Section 2. In Section 3, we present the baseline algorithm and dichotomy-based algorithm. In Section 4, we introduce the index-based algorithm. We report the evaluation of effectiveness and efficiency of our strategies in Section 5. Finally, we show the related work in Section 6 and conclude the study in Section 7.

2. Preliminaries

In this section, we first introduce some necessary concepts and present the formal definition of the personalized influential k-ECC search problem. Table 1 summarizes the notations that are frequently used in this study.

We consider a network $G = (V, E)$ as an undirected graph, where V and E represent the set of vertices and edges in G, respectively. $n = |V|$ and $m = |E|$ are the number of

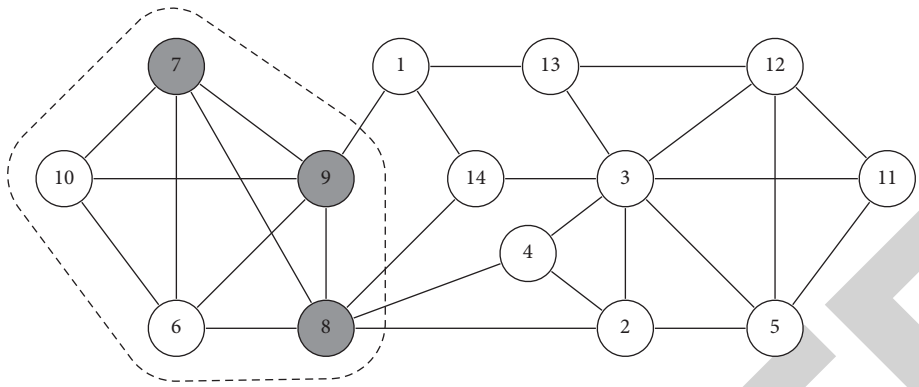

Figure 1: Running example (the number in each vertex denotes its vertex id and weight).

Table 1: Summary of notations.

Notation	Definition
$G = (V, E)$	A graph with vertex/edge set V/E
$g = (V_g, E_g)$	An induced subgraph of G
n	Number of vertices in G
m	Number of edges in G
u, v	Vertex in G
$w(u)$	The influence value of u in G
$f(g)$	The influence value of subgraph g
$\lambda(u, v)$	The connectivity between u and v
$\lambda(g)$	The connectivity of subgraph g
k	The connectivity constraint
Q	The query vertex set

vertices and edges. Each vertex $u \in V$ is associated with a weight $\omega(u)$, representing its influence value. Without loss of generality, we use the same setting as the previous works for vertex weight, where different vertices have different weights [12, 14]. For vertices with the same weight, we break the tie randomly. A subgraph $g = (V_g, E_g)$ is an induced graph of G, if $V_g \subseteq V$ and $E_g = \{(u, v) | u, v \in V_g \wedge (u, v) \in E\}$. To measure the cohesiveness of subgraph, we utilize the k-edge-connected component (k-ECC) model, which is widely adopted [16].

Definition 1 (connectivity). Given a subgraph g and two vertices $u, v \in V_g$, the connectivity $\lambda(u, v)$ between u and v is the minimum number of edges, whose removal will disconnect u and v in g. The connectivity of g is the minimum connectivity between any two distinct vertices in g, i.e., $\lambda(g) = \min_{u,v \in V} \lambda(u, v)$.

Definition 2 (k-ECC). Given a graph G, a subgraph g is a k-edge-connected component (k-ECC) of G if (i) $\lambda(g) \geq k$ and (ii) g is maximal, i.e., the connectivity of any supergraph of g is less than k.

To compute the k-ECCs, we apply the state-of-the-art method, which iteratively decomposes the graph by removing the unpromising edges [16]. As we discussed, we want to identify the community, which is not only cohesive but also has large influence value.

Definition 3 (influence value). Given a subgraph g, the influence value of g is denoted by the minimum weight of the vertex in V_g, i.e., $f(g) = \min_{u \in V_g} \omega(u)$.

In the previous studies, people usually focus on finding all or top-r influential communities, while the property of personalization is ignored, which is an important factor for network analysis. Inspired by this, we formally define the personalized influential k-ECC (PIKE) as follows.

Definition 4 (personalized influential k-ECC). Given a graph G, a positive integer k, and a query vertex set Q, a personalized influential k-ECC (PIKE) is an induced subgraph g of G, which meets all the following constraints:

(i) Personalized: Q is contained in g, i.e., $Q \subseteq V_g$

(ii) Cohesiveness: $\lambda(g) \geq k$

(iii) Maximal: there is no other subgraph g' that (i) satisfies the first two constraints, (ii) is a supergraph of g, i.e., $g' \supset g$, and (iii) has the same influence value as g, i.e., $f(g) = f(g')$

(iv) Largest: g is the one with the largest influence value and satisfies the previous constraints

2.1. Problem Statement. Given a graph $G = (V, E)$, a set Q of query vertices, and a positive integer k, we aim to develop an efficient algorithm to find the PIKE for query vertices Q.

3. Solution

In this section, a baseline algorithm is firstly developed. Novel algorithms are further proposed to accelerate the computation.

3.1. Baseline Algorithm. Before introducing the baseline algorithm, we first present an important property about community influence value.

Lemma 1. *Given a graph G and two induced subgraphs g_1 and g_2, we have $V_{g_1} = V_{g_2} \cup \{u\}$. If the weight of u is smaller than the influence value of g_2, i.e., $\omega(u) < f(g_2)$, then the*

influence value of g_1 is smaller than that of g_2, i.e., $f(g_1) < f(g_2)$.

Proof. Based on the definition of influence value, we have $f(g_1) = \min_{v \in V_{g_1}} \omega(v) \leq \omega(u) < f(g_2)$. Thus, the lemma is correct.

According to Lemma 1, for a given subgraph g, we can increase its influence value by iteratively deleting the vertex with the smallest weight. Algorithm 1 presents the details of baseline method. We first compute the k-ECC g that contains the query vertices Q in Lines 1-2. COMPUTE KECCS is the algorithm developed in [16], which is the state-of-the-art method for k-ECC computation. If the query vertices are not contained in any k-ECC, an error code is returned in Line 3, which means there is no satisfied community for the query. Otherwise, we sort the vertices of g in ascending order based on vertex weights and store the vertices, whose weight is smaller than that of Q, into S. If S is empty, then g is returned. Otherwise, we try to delete the vertex with the current smallest weight one by one in Line 7. For each processed vertex w, deleting it may break the connectivity of other vertices. Hence, we need to make sure the remained subgraph satisfies the connectivity constraint. When there is no k-ECC containing Q, we return the k-ECC g found in the previous iteration as the result (Lines 10-11). Note that, the vertices in S remain sorted as Line 4 does. Based on Lemma 1, the correctness of the algorithm is straightforward. □

Example 2. Considering the graph in Figure 1. Assume $k = 2$ and query vertex set Q is $\{v_5, v_6\}$. By the definition of baseline algorithm, we first compute the k-ECC of G. Then, we iteratively remove the vertex with the current smallest weight, i.e., v_1, v_2, and v_3, and compute the result. After deleting the vertex v_3, vertices v_4, v_{13}, and v_{14} violate the degree constraint and are removed. The remained graph is separated into two connected components. As we can see, there is no 2-ECC containing query set Q. Then, we stop and output 2-ECC containing Q by $\{v_3, v_4, \ldots, v_{14}\}$.

3.2. Dichotomy-Based Algorithm. The core part of the baseline algorithm is to locate the critical vertex, whose removal will lead to none k-ECC (i.e., Lines 10-11 of Algorithm 1). To find the vertex, we need to delete the vertex orderly and check the result accordingly. For each deletion of a vertex, the computation of k-ECC is required, which is time-consuming, especially for large graphs. If we can remove a bulk of candidate vertices at once, it will avoid a lot of computation. Motivated by this, we introduce a dichotomy-based algorithm to accelerate the processing. The details are shown in Algorithm 2.

In Algorithm 2, the first three steps (i.e., Lines 1–4) are the same as the baseline algorithm, which is to find the k-ECC containing Q and initialize the candidate vertex set S. Then, we process the candidates in a dichotomy manner. That is, we iteratively partition the vertices in S into two sets D_1 and D_2. D_1 contains the vertices in the first half of S, and D_2 consists of the others (Lines 6-7). If $|S| = 1$, we put the vertex in D_1. Next, we try to remove the vertices in D_1 from the current k-ECC g and compute the k-ECC on the remained subgraph g' (Lines 8-9). If the returned k-ECC still contains Q, we repeat the procedure on the remained graph (Line 15). Otherwise, it means we can only remove part of vertices in D_1 to obtain the PIKE. The procedure terminates if S is empty or no more vertex can be removed from the candidate set (Lines 5 and 12). By conducting the search in a dichotomy manner, we can reduce the computation time of k-ECC from $O(|S|)$ to $O(\log_2|S|)$, whose advantages could also be observed from our experiment evaluation.

Example 3. Reconsidering the graph in Figure 1. Assume $k = 2$ and query vertex set Q is $\{v_5, v_6\}$. By the definition of dichotomy-based algorithm, we first compute 2-ECC containing Q and then initialize the candidate vertex set S with $\{v_1, v_2, v_3, v_4\}$. We partition the vertices in S into D_1 (i.e., $\{v_1, v_2\}$) and D_2 (i.e., $\{v_3, v_4\}$). Next, we remove the vertices in D_1 and obtain 2-ECC containing Q in the remained subgraph. The candidate set S is updated by $\{v_3, v_4\}$. We repeat the same procedure, and we can find that there is no 2-ECC containing Q after removing D_1 (i.e., $\{v_3\}$). So, we return the result obtained in last iteration, i.e., $\{v_3, v_4, \ldots, v_{14}\}$.

4. Index-Based Algorithm

Compared with the baseline method, the dichotomy-based algorithm can significantly reduce the cost of computing k-ECCs. However, this approach still has some limitations: (i) deleting vertices and recomputing k-ECC still cost a lot for processing large graphs and (ii) in real-world applications, different users may have different requirements and there may exist a lot of queries. Therefore, it will be difficult for the method to meet the online requirements. Motivated by this, we develop an index-based algorithm. The idea is that, for each k, we follow the vertex deletion procedure by iteratively removing the vertex with the smallest weight in current k-ECC. The deletion of certain vertex will cause some other vertices to violate the connectivity constraint and be removed from the k-ECC. We keep the order of these vertices and construct a tree index.

4.1. Index Construction. The index construction details are shown in Algorithm 3. For each k from 1 to $k_{\max}$, we construct a tree index (Line 2). Then, we process each k-ECC of G with Build Node procedure, whose details are in Lines 7-14. In Build Node, we construct an intermediate node for the input k-ECC g. Then, we delete the vertex with the smallest weight in g and compute the k-ECC H on the remained graph (Lines 9-10). We store the vertices violating the connectivity constraint in N and add N as the child node of T (Lines 11-12). The procedure terminates when all the vertices are processed.

Example 4. Figure 2 shows the constructed index for the graph in Figure 1 when $k = 2, 3$. For $k = 3$, the constructed index is shown in Figure 2(b). We first process v_1, which

Input: G: a graph, Q: query vertices, k: a positive integer
Output: PIKE for the query
(1) $H \leftarrow$ Compute KECCs (G, k);
(2) $g \leftarrow$ the k-ECC in H containing Q;
(3) **if** $g = \emptyset$ **then return** error;
(4) $S \leftarrow$ vertices of g with weight smaller than $f(Q)$ and sort vertices in the ascending order according to the weights
(5) **while** $S \neq \emptyset$ **do**
(6) $w \leftarrow S$.front
(7) $g' \leftarrow g \setminus \{w\}$
(8) $H \leftarrow$ Compute KECCs (g', k)
(9) $g' \leftarrow$ the k-ECC in H containing Q
(10) **if** $g' = \emptyset$ **then**
(11) **break**
(12) $S \leftarrow V_{g'}$; $g \leftarrow g'$
(13) **return** g

Algorithm 1: Baseline algorithm

Input: G: a graph, Q: query vertices, k: a positive integer
Output: PIKE for the query
(1) $H \leftarrow$ Compute KECCs (G, k)
(2) $g \leftarrow$ the k-ECC in H containing Q
(3) **If** $g = \emptyset$ **then return** error
(4) $S \leftarrow$ vertices of g with weight smaller than $f(Q)$ and sort vertices in the ascending order according to the weights
(5) **while** $S \neq \emptyset$ **do**
(6) $D_1 \leftarrow$ the first half of vertices in S
(7) $D_2 \leftarrow$ the second half of vertices in S
(8) $g' \leftarrow$ remove D_1 from g
(9) $H \leftarrow$ Compute KECCs (g', k)
(10) $g' \leftarrow$ the k-ECC in H containing Q
(11) **if** $g' = \emptyset$ **then**
(12) **if** $|D_1| = 1$ **then break;**
(13) $S \leftarrow D_1$
(14) **else**
(15) $S \leftarrow V_{g'} \cap D_2$; $g \leftarrow g'$
(16) **return** g

Algorithm 2: Dichotomy-based algorithm

Input: G: a graph
Output: constructed Index
(1) **for** k from 1 to $k_{\max}$ do
(2) $T_k \leftarrow$ initialize a tree root node for k
(3) $H \leftarrow$ Compute KECCs (G, k)
(4) **for each** $g \in H$ **do**
(5) Build Node (T_k, g, k)
(6) **return** $T_1, T_2, \ldots, T_{\max}$
(7) **Procedure** Build Node (T, g, k)
(8) construct a tree node N
(9) $u \leftarrow$ the vertex with the smallest weight in g
(10) $H \leftarrow$ Compute KECCs $(g, \{u\}, k)$
(11) $N \leftarrow V_g$, vertices in H
(12) add N as a child node of T
(13) **for each** $g' \in H$ do
(14) Build Node (N, g', k)

Algorithm 3: Index construction

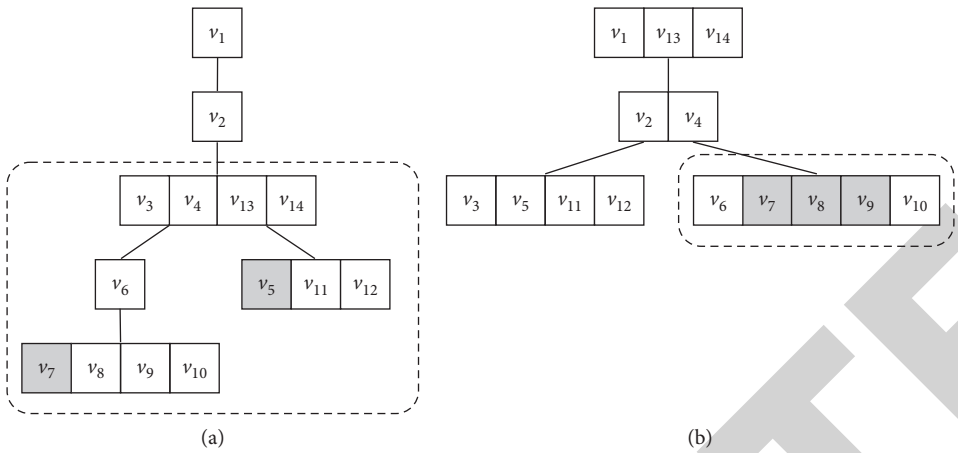

Figure 2: Index construction and query processing example: (a) $k = 2$ and (b) $k = 3$.

leads to the removal of v_{13} and v_{14}. Then, we remove v_2 and cause v_4 to violate the connectivity constraint. Deleting v_4 results in 2 connected components. When deleting v_3, the connectivity of v_5, v_{11}, and v_{12} becomes less than 3. Therefore, we construct a tree node for the four vertices. For the other connected component, we conduct similar procedure and the constructed index is shown in the right branch.

4.2. Query Processing. For a given query, we first retrieve the tree index by k. Then, we locate the intermediate tree nodes that contain the query vertices. Finally, we only need to find their closest common ancestor N_a and return all the vertices in N_a and its child nodes as the results. Following is an example of index construction and query processing.

Example 5. For $k = 2$, we perform the similar procedure, where the corresponding index is shown in Figure 2(a). Given a query with $k = 2$ and $Q = \{v_5, v_7\}$, then we retrieve the closest common ancestor node of v_5, v_7. Therefore, the vertices in the dotted circle of Figure 2(a) are the results. Similarly, when $k = 3$ and $Q = \{v_7, v_8, v_9\}$, the vertices in the dotted circle of Figure 2(b) are the results.

5. Experiment

5.1. Algorithms. To the best of our knowledge, there is no existing work for the proposed problem. In the experiments, we implement and evaluate the following algorithms:

(i) BL: the baseline algorithm proposed in Algorithm 1.

(ii) DBA: the dichotomy-based algorithm proposed in Algorithm 2.

(iii) IBA: the index-based algorithm proposed in this study.

5.2. Datasets and Workloads. We employ 6 real-world networks, which are publicly available on SNAP (http://snap.stanford.edu). Their number of vertices and edges is reported in Table 2. We use the PageRank score of vertex to denote its weight, which is widely used in existing studies [12, 14]. To evaluate the performance of proposed techniques, we vary the parameter k, the number of query vertices $|Q|$, and the weight distribution ω of query vertices. For each setting, we randomly generate 20 queries with nonempty results and run the algorithms 10 times to report average response time. All algorithms are implemented in C++, and all the experiments are performed on a PC with an Intel i5-9600KF CPU and 32 GB RAM.

5.3. Results of Varying k. We first conduct the experiments on all datasets by varying k from 5 to 25. For each query Q, we randomly selected 10 vertices from the graph. The corresponding results are shown in Figures 3(a)–3(f). We can see that IBA and DBA significantly outperform BL by a wide margin. In Gowalla, IBA can achieve up to 7 orders of magnitude speedup compared with the baseline method. This is because, with the proposed index, we only need to access the data that are related to queries. As observed, when k increases, the running time decreases for all methods, since the community size decreases and more space could be directly pruned based on the cohesive subgraph model.

5.4. Results of Varying |Q|. To further evaluate the performance, we vary the number $|Q|$ of queried vertices from 2 to 30 with $k = 15$ as default. The results are shown in Figures 4(a)–4(f), where similar trends can be observed. IBA is significantly faster than BL because of the index structure developed. The running time of BL and DBA decreases when $|Q|$ increases. This is because larger $|Q|$ will lead to smaller size of candidate vertices in S. The running time of IBA slightly increases when $|Q|$ increases because it will need more time to locate the corresponding tree nodes and their common ancestor.

5.5. Results of Varying ω. We sort the vertices in an increasing order according to the weights and divide them into 5 buckets. We vary the weight distribution ω of queried vertices from 20% to 100%. The results are shown in

Table 2: Statistics of datasets.

Dataset	$n = \|V\|$	$m = \|E\|$
Email-Eu-core	1,005	16,064
Facebook	4,039	88,234
Email-Enron	36,692	367,662
Brightkite	58,228	214,078
Gowalla	196,591	950,327
YouTube	1,134,890	2,987,624

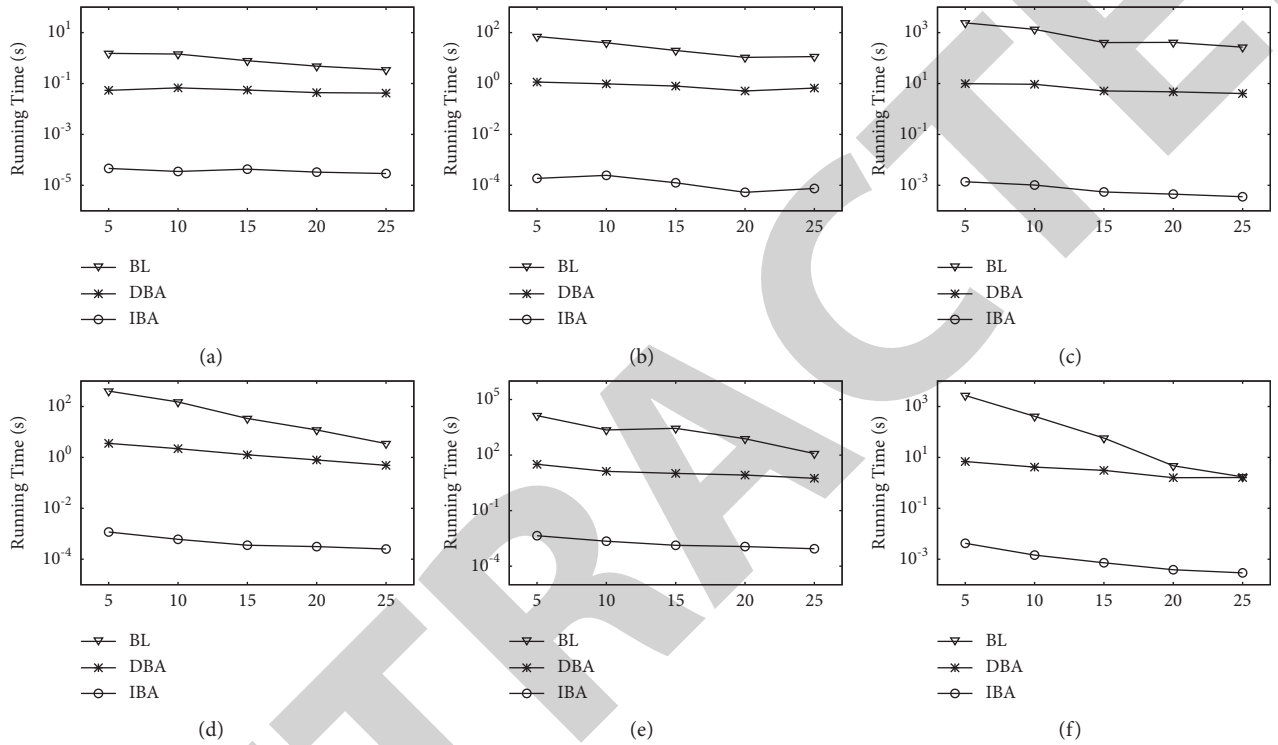

Figure 3: Experiment results by varying parameters k: (a) Email-Eu-core, (b) Facebook, (c) Email-Enron, (d) Brightkite, (e) Gowalla, and (f) YouTube.

Figures 5(a)–5(f), where x% means the query vertices are selected from the range x% to $(x + 20)$%. Larger x% means higher weight. Note that, to make it fair, for each query, we set $|Q| = 2$ and $k = 5$. This is because, for larger $|Q|$ and k, it may lead to lots of empty results. As shown, DBA and IBA are not sensitive to ω. When ω increases, the response time of BL increases greatly. This is because, larger ω means larger candidate size, i.e., $|S|$. Since BL processes the candidates in linear manner, it will invoke the k-ECC computation procedure a lot of times. In Gowalla, when $\omega = 80\%$, BL requires 64567.46 s to find the result, while IBA only takes 0.00139 s because of the advanced index developed in this study.

6. Related Work

6.1. Cohesive Subgraph Mining. In the literature, computing cohesive subgraphs has been widely studied, where different models are proposed to measure the cohesiveness of community, such as k-core [23, 24], k-ECC [16], k-truss [25, 26], and clique [27, 28]. These works aim to compute all maximal subgraphs whose cohesiveness is no smaller than a given threshold. In [16–18], novel techniques are developed to identify cohesive subgraphs based on the k-ECC model. Compared with the k-core model, the k-ECC model enjoys much more cohesiveness. In general, there are three methods to compute k-ECCs of a graph, i.e., cut-based method [29], decomposition-based method [16], and random contraction [17]. However, these techniques cannot be applied to the problem studied in this study due to different problem definitions.

6.2. Influential Community Detection. As discussed, users in the social networks are usually associated with weights denoting their influence. Reference [12] presents a novel model, named k-influential community, based on the k-core concept, and tries to find the top-r k-influential communities from the networks. Considering the importance of the problem, in [13], a backward search algorithm is presented to enable early termination. Moreover, in [14], a local search algorithm is developed to overcome the deficiency of accessing the whole

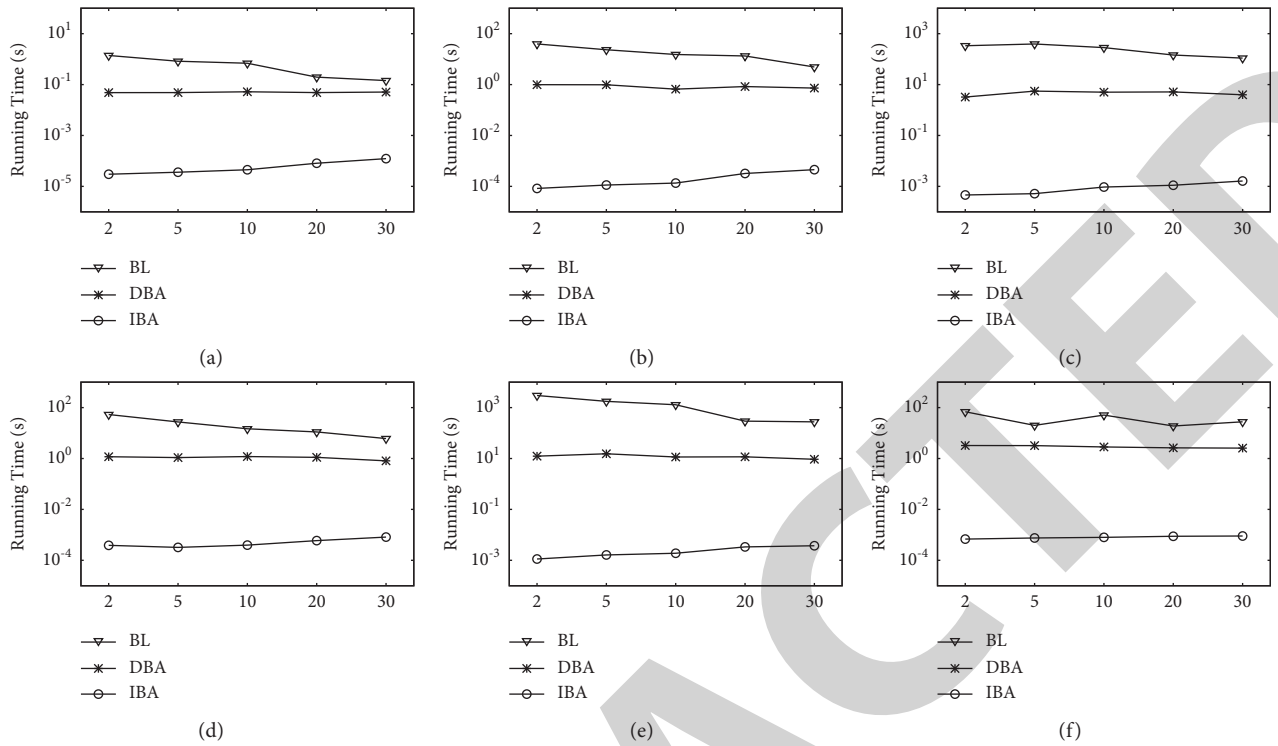

Figure 4: Experiment results by varying number of query vertices $|Q|$: (a) Email-Eu-core, (b) Facebook, (c) Email-Enron, (d) Brightkite, (e) Gowalla, and (f) YouTube.

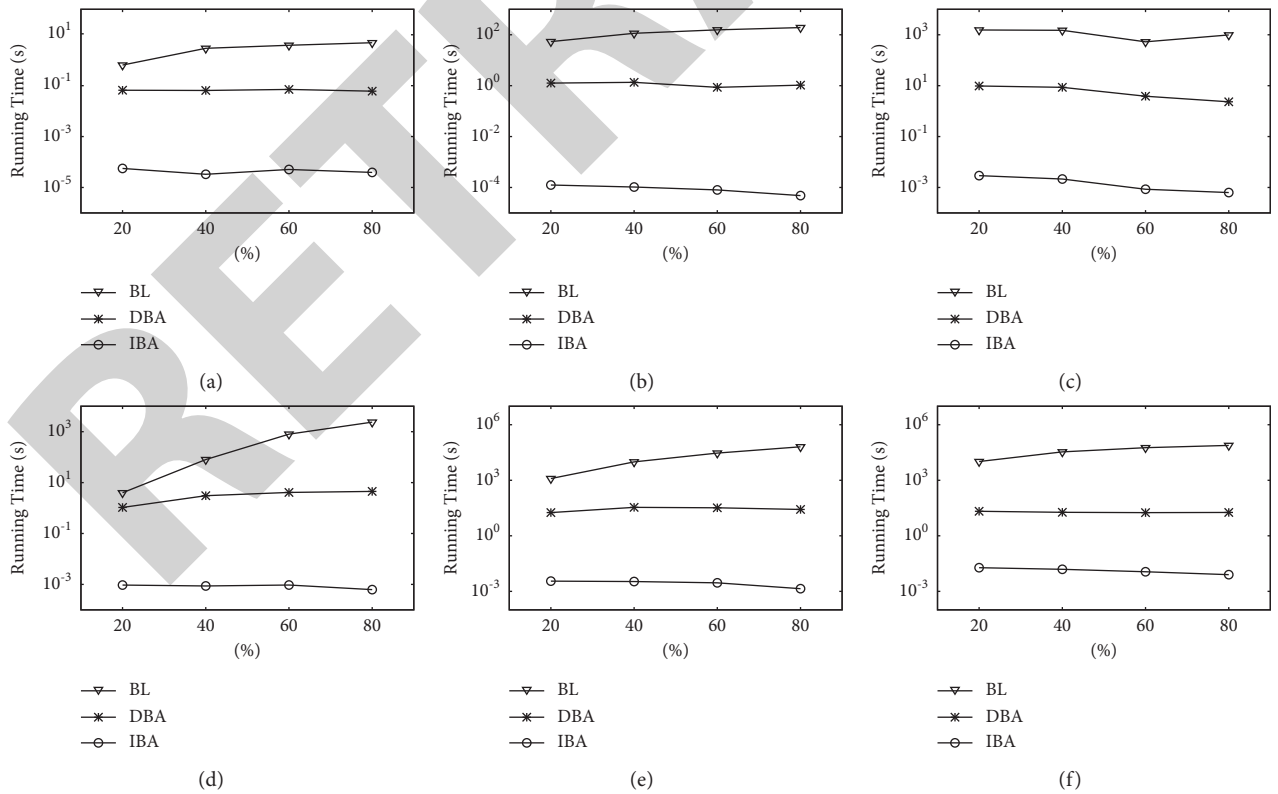

Figure 5: Experiment results by varying weight distribution ω of query vertices: (a) Email-Eu-core, (b) Facebook, (c) Email-Enron, (d) Brightkite, (e) Gowalla, and (f) YouTube.

graph. In [15], a personalized influential community search has been proposed, which aims to retrieve the most influential community for query vertex by leveraging the k-core concept. In our previous work [19], a k-ECC-based community search model is proposed. However, it only focuses on the community with maximum k instead of any given k as in this study.

6.3. Community Search. Community search, which aims to find cohesive subgraphs containing the query vertices, has been widely studied (e.g., [10, 30]). Refs. [31, 32] use the minimum degree to server as the metric to measure the cohesiveness of a community and aim to find the maximal connected k-core with maximal k value. Reference [32] proposes a global search algorithm, and Reference [31] proposes a local search method for the problem. In [33], the authors study the online community search based on the k-truss concepts and develop a novel tree shape index, i.e., TCP index, to efficiently search k-truss community. There are many studies on other types of graphs, e.g., attribute graphs and signed graphs [34, 35]. A comprehensive survey about recent studies on community search problem can be found in [7]. As observed, there is a lack of research for personalized influential community search problem, which is of great importance for many social network-based applications.

7. Conclusion

Graphs are widely used to model the complex relationships among different entities. In graph analysis, community search is a fundamental problem and receives great attention recently. In a network, users are usually associated with weights denoting its influence in the network, which is neglected by most previous studies. In this study, we conduct the first research to investigate the personalized influential k-ECC (PIKE) search problem in large networks. We formally define the problem and propose a baseline algorithm. To reduce the cost of k-ECC computation, a dichotomy-based algorithm is developed to reduce the searching space. In real scenarios, there may be a lot of queries issued. To meet the online requirement in real applications, an index-based algorithm is further developed to accelerate the computation. Experiments are conducted on 6 real-world social networks to verify the advantages of proposed techniques.

Data Availability

The datasets used in the study are publicly available at https://snap.stanford.edu/data/index.html.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

This work was supported by ZJECF Y201839942, ZJECF Y202045024, NSFC 61802345, ZJNSF LQ20F020007, and ZJNSF LY21F020012.

References

[1] H. Li, T. Wang, W. Pan et al., "Mining key classes in java projects by examining a very small number of classes: a complex network-based approach," *IEEE Access*, vol. 9, pp. 28076–28088, 2021.

[2] X. Wang, Y. Zhang, W. Zhang, and X. Lin, "Efficient distance-aware influence maximization in geo-social networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 3, pp. 599–612, 2017.

[3] X. Wang, Y. Zhang, W. Zhang, X. Lin, and C. Chen, "Bring order into the samples: a novel scalable method for influence maximization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 2, pp. 243–256, 2017.

[4] X. Liu and X. Wang, "Cohesive subgraph identification in weighted bipartite graphs," *Applied Sciences*, vol. 11, no. 19, p. 9051, 2021.

[5] X. Liu, M. Zhang, X. Fu, C. Chen, X. Wang, and Y. Wu, "Efficient reachability queries in multi-relation graph: an index-based approach," *Computers & Electrical Engineering*, vol. 96, p. 107469, 2021.

[6] X. Huang, V. Laks, S. Lakshmanan, and J. Xu, "Community search over big graphs: models, algorithms, and opportunities," in *Proceedings of the 33rd IEEE International Conference on Data Engineering, ICDE 2017*, pp. 1451–1454, IEEE Computer Society, San Diego, CA, USA, April 2017.

[7] Y. Fang, X. Huang, L. Qin et al., "A survey of community search over big graphs," *The VLDB Journal*, vol. 29, no. 1, pp. 353–392, 2020.

[8] H. Steve, B. Gonzalo, L. Gjeltema et al., "Community detection in large-scale networks: a survey and empirical evaluation," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 6, no. 6, pp. 426–439, 2014.

[9] A. Conte, T. De Matteis, D. De Sensi, R. Grossi, A. Marino, and L. Versari, "D2K: scalable community detection in massive networks via small-diameter k-plexes," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018*, pp. 1272–1281, ACM, London, UK, August 2018.

[10] Y. Fang, R. Cheng, S. Luo, and J. Hu, "Effective community search for large attributed graphs," *Proceedings of the VLDB Endowment*, vol. 9, no. 12, pp. 1233–1244, 2016.

[11] Z. Chen, H. Lu, S. Tian et al., "Construction of a hierarchical feature enhancement network and its application in fault recognition," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4827–4836, 2020.

[12] R.-H. Li, L. Qin, J. X. Yu, and R. Mao, "Influential community search in large networks," *Proceedings of the VLDB Endowment*, vol. 8, no. 5, pp. 509–520, 2015.

[13] S. Chen, W. Ran, D. Popova, and A. Thomo, "Efficient computation of importance based communities in web-scale networks using a single machine," in *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016*, Indianapolis, IN, USA, October 2016.

[14] F. Bi, L. Chang, X. Lin, and W. Zhang, "An optimal and progressive approach to online search of top-k influential communities," *Proceedings of the VLDB Endowment*, vol. 11, no. 9, 2018.

[15] Y. Wu, J. Zhao, R. Sun, C. Chen, and X. Wang, "Efficient personalized influential community search in large networks," in *Proceedings of the Web and Big Data-4th International Joint Conference, APWeb-WAIM 2020*, pp. 86–101, Springer, Tianjin, China, September 2020.

[16] L. Chang, J. Xu Yu, Q. Lu, X. Lin, C. Liu, and W. Liang, "Efficiently computing k-edge connected components via graph decomposition," in *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013*, New York, NY, USA, June 2013.

[17] T. Akiba, Y. Iwata, and Y. Yoshida, "Linear-time enumeration of maximal k-edge-connected subgraphs in large networks by random contraction," in *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management, CIKM'13*, San Francisco, CA, USA, October 2013.

[18] L. Chang, X. Lin, Q. Lu, J. Xu Yu, and W. Zhang, "Index-based optimal algorithms for computing steiner components with maximum connectivity," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, Melbourne, Australia, May 2015.

[19] Y. Xu, R. Sun, Y. Wu, C. Chen, and X. Wang, "Querying influential maximum connected community in large graphs," in *Proceedings of the 2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 757-758, Sydney, Australia, October 2020.

[20] X. Yan, X. Jasmine Zhou, and J. Han, "Mining closed relational graphs with connectivity constraints," in *Proceedings of the 21st International Conference on Data Engineering, ICDE 2005*, pp. 357-358, IEEE Computer Society, Tokyo, Japan, April 2005.

[21] C. Dorn and S. Dustdar, "Composing near-optimal expert teams: a trade-off between skills and connectivity," in *Proceedings of the On the Move to Meaningful Internet Systems: OTM 2010-Confederated International Conferences: CoopIS, IS, DOA and ODBASE*, pp. 472–489, Springer, Hersonissos, Greece, October 2010.

[22] A. Beutel, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos, "Copycatch: stopping group attacks by spotting lockstep behavior in social networks," in *Proceedings of the 22nd International World Wide Web Conference, WWW' 13*, Rio de Janeiro, Brazil, May 2013.

[23] C. Chen, Q. Zhu, R. Sun, X. Wang, and Y. Wu, "Edge manipulation approaches for k-core minimization: metrics and analytics," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[24] R. Sun, C. Chen, X. Wang, Y. Zhang, and X. Wang, "Stable community detection in signed social networks," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[25] J. Zhao, R. Sun, Q. Zhu, X. Wang, and C. Chen, "Community identification in signed networks: a k-truss based model," in *Proceedings of the CIKM' 20: the 29th ACM International Conference on Information and Knowledge Management*, pp. 2321–2324, Virtual Event, Ireland, October 2020.

[26] W. Zhu, M. Zhang, C. Chen, X. Wang, F. Zhang, and X. Lin, "Pivotal relationship identification: the k-truss minimization problem," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019*, Macao, China, August 2019.

[27] C. Chen, Y. Wu, R. Sun, and X. Wang, "Maximum signed θ-clique identification in large signed graphs," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[28] Z. Tang, H. Hu, C. Xu, and K. Zhao, "Exploring an efficient remote biomedical signal monitoring framework for personal health in the covid-19 pandemic," *International Journal of Environmental Research and Public Health*, vol. 18, no. 17, p. 9037, 2021.

[29] X. Yan, X. Jasmine Zhou, and J. Han, "Mining closed relational graphs with connectivity constraints," in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Chicago, IL, USA, August 2005.

[30] X. Huang, V. S. Lakshmanan, and J. Xu, "Community search over big graphs: models, algorithms, and opportunities," in *Proceedings of the 33rd IEEE International Conference on Data Engineering, ICDE 2017*, San Diego, CA, USA, April 2017.

[31] W. Cui, Y. Xiao, H. Wang, and W. Wang, "Local search of communities in large graphs," in *Proceedings of the International Conference on Management of Data, SIGMOD 2014*, Snowbird, UT, USA, June 2014.

[32] M. Sozio and A. Gionis, "The community-search problem and how to plan a successful cocktail party," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, DC, USA, July 2010.

[33] X. Huang, H. Cheng, Q. Lu, W. Tian, and J. Xu Yu, "Querying k-truss community in large and dynamic graphs," in *Proceedings of the International Conference on Management of Data, SIGMOD 2014*, Snowbird, UT, USA, June 2014.

[34] R. Sun, C. Chen, X. Wang, Y. Wu, M. Zhang, and X. Liu, "The art of characterization in large networks: finding the critical attributes," *World Wide Web Journal*, pp. 1–23, 2021.

[35] R. Sun, Q. Zhu, C. Chen, X. Wang, Y. Zhang, and X. Wang, "Discovering cliques in signed networks based on balance theory," in *Proceedings of the Database Systems for Advanced Applications-25th International Conference, DASFAA 2020*, Jeju, South Korea, September 2020.