World Wide Web (2021) 24:397–417
https://doi.org/10.1007/s11280-020-00843-6

Parallel algorithms for parameter-free structural diversity search on graphs

Jinbin Huang[1] · Xin Huang[1] · Yuanyuan Zhu[2] · Jianliang Xu[1]

Received: 21 March 2020 / Revised: 22 July 2020 / Accepted: 21 September 2020 /
Published online: 20 November 2020

Abstract
Structural diversity of a user in a social network is the number of social contexts in his/her contact neighborhood. The problem of structural diversity search is to find the top-k vertices with the largest structural diversity in a graph. However, when identifying distinct social contexts, existing structural diversity models (e.g., t-sized component, t-core, and t-brace) are sensitive to an input parameter of t. To address this drawback, we propose a parameter-free structural diversity model. Specifically, we propose a novel notation of discriminative core, which automatically models various kinds of social contexts without parameter t. Leveraging on discriminative cores and h-index, the structural diversity score for a vertex is calculated. We study the problem of parameter-free structural diversity search in this paper. An efficient top-k search algorithm with a well-designed upper bound for pruning is proposed. To further speed up the computation, we design a novel parallel algorithm for efficient top-k search over large graphs. The parallel algorithm computes diversity scores for a batch of vertices simultaneously using multi-threads. Extensive experiment results demonstrate the parameter sensitivity of existing t-core based model and verify the superiority of our methods.

Keywords Structural diversity search · Parallelization · Parameter-free · H-index

This article belongs to the Topical Collection: *Special Issue on Web Information Systems Engineering 2019*
Guest Editors: Reynold Cheng, Nikos Mamoulis, and Xin Huang

✉ Xin Huang
xinhuang@comp.hkbu.edu.hk

Jinbin Huang
jbhuang@comp.hkbu.edu.hk

Yuanyuan Zhu
yyzhu@whu.edu.cn

Jianliang Xu
xujl@comp.hkbu.edu.hk

[1] Hong Kong Baptist University, Kowloon Tong, Hong Kong

[2] Wuhan University, Wuhan, China

1 Introduction

Nowadays, information spreads quickly and widely on social networks (e.g., Twitter, Facebook) [24, 29, 35, 47]. Individuals are usually influenced easily by the information received from their social neighborhoods. Recent studies show that social decisions made by individuals often depend on the multiplicity of social contexts inside his/her contact neighborhood, which is termed as *structural diversity* [22, 47]. Individuals with larger structural diversity, are shown to have higher probability to be affected in the process of social contagion [47]. Structural diversity search, finding the individuals with the highest structural diversity in graphs, has many applications such as political campaigns [26], viral marketing [30], promotion of health practices [47], facebook user invitations [47], twitter user retention [45], social reputation evaluation [53] and so on.

In the literature, several structural diversity models (e.g., t-sized component, t-core and t-brace) need an input of specific parameter t to model distinct social contexts. A social context is formed by a number of connected users. The component-based structural diversity [47] regards each connected component whose size is larger than t as a social context. Another core-based structural diversity model is defined based on t-core. A t-core is the largest subgraph such that each vertex has at least t neighbors within t-core. The core-based structural diversity model regards each maximal connected t-core as a distinct social context. Figure 1 shows the contact neighborhood (ego-network) $G_{N(v)}$ of a user v. All vertices and edges in ego-network $G_{N(v)}$ are in solid lines. Consider the core-based structural diversity model and parameter $t = 2$. Subgraphs H_1, H_2 and H_3 are maximal connected 2-cores. H_1, H_2, and H_3 are regarded as 3 distinct social contexts. Thus, the core-based structural diversity of v is 3.

This paper proposes a new parameter-free structural diversity model based on the core-based model [21] and h-index measure [18]. Our parameter-free model does not need the input of parameter t any more. This avoids suffering from the limitations of setting parameter t. We show two major drawbacks of the t-core based model as follows.

– **Sensitivity of t-core based model**. The number of social contexts is sensitive to parameter t. On the one hand, if t is set to a large value, it may discard small and weakly-connected social contexts; On the other hand, if t is set to a small value, it may have weak ability of recognizing strongly-connected social contexts fully. Consider the contact neighborhood $G_{N(v)}$ of a user v in Figure 1. When $t = 2$, the structural diver-

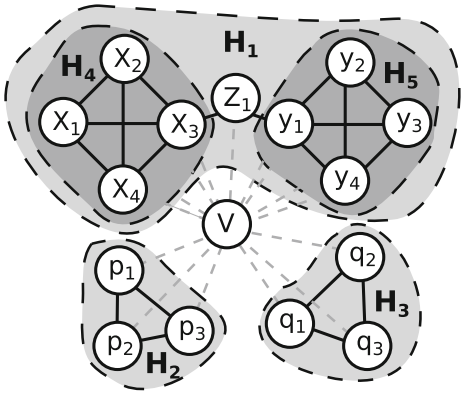


Figure 1 The ego-network $G_{N(v)}$ of vertex v

sity of v is 3. When $t = 3$, H_2 and H_3 are 2-cores and disqualified for social contexts, due to the requirement of social contexts as 3-core. Meanwhile, H_1 is decomposed as two components of 3-core as H_4 and H_5. Thus, the structural diversity of v becomes 2. However, when $t \geq 4$, the structural diversity of v is 0. This example clearly shows the sensitivity of structural diversity w.r.t. parameter t.

– **Inflexibility of t-core based model**. Structural diversity model lacks flexibility for different vertices using the same parameter t. Generally, different social contexts should not be modeled and quantified using the same criteria of parameter t. For example, in a social network, the social contexts of a famous singer and a junior student can be dramatically different in terms of size and density. Thus, it is difficult to choose one consistent value t for different vertices in a graph. In Figure 1, H_1 can be decomposed into two social contexts H_4 and H_5, which requires the setting of $t = 3$. However, the identification of H_2 and H_3 requires $t = 2$. This indicates the necessary of personalized parameter t for different social contexts.

To address the above two limitations, we define a novel notation of discriminative core to represent each distinct social context without inputing any parameters. Specifically, a discriminative core is a densest and maximal connected subgraph inside a user's contact neighborhood. It can be regarded as a criteria for representing unique and strong social context. However, the distribution of discriminative cores in two users' contact neighborhoods can be totally different in terms of density and quantity, which cannot be compared directly. To tackle this issue, we propose a new structural diversity model based on h-index. In the literature, the h-index is defined as the maximum number of h such that a researcher has published h papers whose citations have at least h [18]. We apply the similar idea to measure structural diversity in ego-networks. Given a vertex v, the structural diversity of v is the largest number h such that there exist at least h discriminative cores with the coreness of at least h.

In this paper, we study the problem of top-k h-index based structural diversity search, which finds k vertices with the largest h-index based structural diversity. To address the problem, we first propose a baseline algorithm to compute the structural diversity scores for all vertices in a graph. In order to improve the top-k search efficiency, we develop an upper bound of structural diversity to prune the vertices whose structural diversity scores certainly absent in the results. Leveraging this pruning technique, our efficient top-k search framework can reduce the search space and terminate in a faster way.

Over the conference version [23] of this manuscript, we further study a parallel algorithm for h-index based structural diversity search over large graphs in Section 6, where multiple threads are utilized for parallel computing in single machine. The motivations are intuitive. A commodity machine usually has multiple processors nowadays, top-k search algorithms run in the single thread, which wastes the computing resources and makes utilization in low efficiency. Moreover, we observe that the computations of upper bounds and structural diversity scores for different vertices are independent, which can be extended to compute in parallel. Thus, we further accelerate the structural diversity search using multi-threading techniques. Although the parallel computation of diversity scores is straightforward, it is difficult to make use of the existing pruning strategy to terminate the algorithm early. Thus, the parallel efficiency is still bottlenecked, due to quite a large workload of score computations for all vertices. To address this issue, we design a new parallel scheme for top-k structural diversity search, in which the search space can be pruned efficiently. Specifically, we develop parallel computations for estimating upper bounds and calculating diversity scores. To ensure the answer correctness, we propose a new vertex search order and

synchronous computations for ranking vertices and updating top-k answers. All these strategies ensure the high efficiency and correct answers for our parallel algorithm.

In summary, we make the following contributions:

- We propose a novel definition of discriminative core to provide a parameter-free scheme for identifying social contexts. To simultaneously measure the quantity and strength of social contexts in one's contact neighborhood, we propose a new h-index based structural diversity model. We formulate the problem of top-k h-index based structural diversity search in a graph (Section 3).
- We propose a useful approach for computing the h-index based structural diversity score $h(v)$ for a vertex v and give a baseline algorithm for solving the top-k structural diversity search problem (Section 4).
- Based on the analysis of the discriminative core structure and the property of h-index, we design an upper bound of $h(v)$. Equipped with the upper bound, we propose an efficient top-k search framework to improve the efficiency (Section 5).
- To further speed up the computation, we design a novel parallel algorithm for efficient top-k structural diversity search in a multiprocessor shared memory machine (Section 6).
- We conduct extensive experiments on four real-world large datasets to demonstrate the parameter sensitivity of the existing core-based structural diversity model and verify the effectiveness of our proposed model. Moreover, we implement the parallel algorithm for top-k structural diversity search in multi-threads and validate the high-performance efficiency, which achieves up to 13X speedup on large Orkut graph. Experiment results also validate the efficiency of our proposed algorithms (Section 7).

2 Related work

This work is related to the studies of structural diversity search, k-core mining, and parallel and distributed graph analytic.

Structural diversity search In [47], Ugander et al. study the structural diversity models in the real-world applications of social contagion. Recently, Su et al. [45] conduct experimental studies on the Twitter platform to investigate the role of structural diversity on retention. In [53], Zhang et al. show that the social reputation of a user is highly controlled by his/her structural diversity. The problem of top-k structural diversity search is proposed and studied by Huang et al. [20, 21]. The goal of the problem is to find k vertices with the highest structural diversity scores. Two structural diversity models based on t-sized component and t-core respectively are studied w.r.t. a parameter threshold t. By improving the efficiency and scalability of the methods [20], Chang et al. [6] propose fast algorithms to address structural diversity search. Recently, Zhang et al. [54] propose an edge based structural diversity model for structural diversity search. Huang et al. [25] develop index-based approaches for top-k truss-based structural diversity search. Besides the structural diversity in social contagion, other structural diversity concepts are proposed for different applications. Cheng et al. [7] propose an approach of diversity-based keyword search to solve the mashup construction problem. Sanz-Cruzado [40] et al. propose a network-based structural

diversity for evaluating the effectiveness of link prediction. Different from above studies, we propose a parameter-free structural diversity model based on the novel definition of discriminative cores, which avoids suffering from the difficulties of parameter tuning. Comparing to the conference paper in [23], in this paper, we design an novel parallelization scheme to accelerate the computation of top-k structural diversity search.

K-core mining There exists lots of studies on k-core mining in the literature. k-core is a definition of cohesive subgraph, in which each vertex has degree at least k [5]. The task of core decomposition is finding all non-empty k-cores for all possible k's. Batagelj et al. [3] proposed an in-memory algorithm of core decomposition. Core decomposition has also been widely studied in different computing environment such as external-memory algorithms [8], streaming algorithms [41], distributed algorithms [38], and I/O efficient algorithms [50]. The study of core decomposition is also extended to different types of graphs such as dynamic graphs [1, 27], uncertain graphs [4], directed graphs [13, 33], temporal graphs [51], and multi-layer networks [16]. Recently, core maintenance in dynamic graphs has attracted significant interest in the literature [1, 34, 52]. In addition, several k-core based community models have been proposed for community search [12, 14, 15, 36].

Parallel and distributed graph analytic In recent years, graph analytic algorithms have been widely studied on a multi-core single machine and distributed computing systems. There exist several studies that focus on developing scalable computing frameworks for graph processing [39, 42, 43]. However, these parallelization frameworks usually target at general purposes for flexibly adapting various graph algorithms. On the other hand, many studies aim at designing parallelization algorithms for tackling specific graph problems, such as pagerank computing [49], triangle listing [2, 48], connected components computing [19, 31], core decomposition [11, 27, 38], truss decomposition [44], and so on. Different from the above studies, we apply parallel techniques on the problem of structural diversity search for finding vertices with the largest structural diversity.

3 Problem statement

In this section, we formulate the problem of h-index based structural diversity search.

3.1 Preliminaries

We consider an undirected and unweighted simple graph $G = (V, E)$, where V is the set of vertices and E is the set of edges. We denote $n = |V|$ and $m = |E|$ as the number of vertices and edges in G respectively. W.l.o.g. we assume the input graph G is a connected graph, which implies that $m \geq n - 1$. For a given vertex v in a subgraph H of G, we define $N_H(v) = \{u \text{ in } H : (u, v) \in E(H)\}$ as the set of neighbors of v in H, and $d_H(v) = |N_H(v)|$ as the degree of v in H. We drop the subscript of $N_G(v)$ and $d_G(v)$ if the context is exactly G itself, i.e. $N(v)$, $d(v)$. The maximum degree of graph G is denoted by $d_{max} = \max_{v \in V} d_G(v)$.

Given a subset of vertices $S \subseteq V$, the subgraph of G induced by S is denoted by $G_S = (S, E(S))$, where the edge set $E(S) = \{(u, v) \in E : u, v \in S\}$. Based on the definition of induced subgraph, we define the ego-network [10, 37] as follows.

Definition 1 (Ego-network) Given a vertex v in graph G, the ego-network of v is the induced subgraph of G by its neighbors $N(v)$, denoted by $G_{N(v)}$.

In the literature, the term "neighborhood induced subgraph" [21] is also used to describe the ego-network of a vertex. For example, consider the graph G in Figure 1. The ego-network of vertex v is shown in the gray area of Figure 1, which excludes v itself with its incident edges. The t-core of a graph G is the largest subgraph of G in which all the vertices have degree at least t. However, the t-core of a graph can be disconnected, which may not be suitable to directly depict social contexts. Hence, we define the connected t-core as follows.

Definition 2 (Connected t-Core) Given a graph G and a positive integer t, a subgraph $H \subseteq G$ is called a connected t-Core iff H is connected and each vertex $v \in V(H)$ has degree at least t in H.

Given a parameter t, the core-based structural diversity model treats each maximal connected t-core as a distinct social context [21, 47]. To measure the structural diversity of an ego-network, one essential step is to tune a proper value for parameter t. However, such parameter setting is not easy and even critically challenging. The following example illustrates it.

Example 1 Figure 1 shows an ego-network $G_{N(v)}$ of vertex v. Given an integer $t = 2$, three maximal connected 2-core (H_1, H_2 and H_3) will be treated as distinct social contexts. The core-based structural diversity of v is 3. When we set $t = 3$, the core-based structural diversity of v will be 2, since H_4 and H_5 will be treated as two distinct social contexts. In this case, H_2 and H_3 are no longer treated as social contexts. If we set t to be some values higher than 3, no social contexts can be identified. The core-based structural diversity of v will then be 0. From this example, we can see that if the value of t is tuned too high, no social contexts can be identified. But if the value of t is set too low, some strong social contexts with denser structures cannot be captured. Thus, to choose a proper value of t for all vertices in a graph is a challenging task.

To tackle the above issue, we propose a parameter-free scheme for automatically identifying strong social contexts in one's ego-network. We firstly give a novel definition of discriminative core based on the concept of coreness as follows.

Definition 3 (Coreness) Given a subgraph $H \subseteq G$, the coreness of H is the minimum degree of vertices in H, denoted by $\varphi(H) = \min_{v \in H}\{d_H(v)\}$. The coreness of a vertex $v \in V(G)$ is $\varphi_G(v) = \max_{H \subseteq G, v \in V(H)}\{\varphi(H)\}$.

Definition 4 (Discriminative Core) Given a graph G and a subgraph $H \subseteq G$, H is a discriminative core if and only if H is a maximal connected subgraph such that there exists no subgraph $H' \subseteq H$ with $\varphi(H') > \varphi(H)$.

By Definition 4, a discriminative core H is a maximal connected component that cannot be further decomposed into smaller subgraphs with a higher coreness. It indicates that a discriminative core is the densest and most important component of a social context, which can be used as a distinct element to represent a social context. In addition, the coreness of

a discriminative core reflects the strength of its representative social context. For example, H_4 is a discriminative core with $\varphi(H_4) = 3$. And H_2 is another discriminative core with $\varphi(H_2) = 2$. According to the core-based structural diversity, they cannot be identified as distinct social contexts simultaneously using the same value of parameter t. But by our discriminative core definition, they will be treated as distinct social contexts automatically without loosing the information of their strength.

For an ego-network $G_{N(v)}$, the whole network may consist of multiple discriminative cores with various corenesses, which can be depicted as a coreness distribution of discriminative cores. Moreover, to rank the structural diversity of two vertices, it is difficult to directly compare the coreness distributions of two ego-networks. Because it is not easy to measure both the number of social contexts and the strength of social contexts simultaneously.

Making use of the idea of h-index criteria, we define the diversity vector and diversity score as follows.

Definition 5 (Diversity Vector and Diversity Score) Given a graph G and a vertex v, the diversity vector of v is the coreness distribution of discriminative cores in $G_{N(v)}$, denoted by $\mathcal{C}(v) = [c_v(1), ..., c_v(n)]$, where $c_v(r) = |\{H : \varphi(H) = r \text{ and } H \text{ is a discriminative core in } G_{N(v)}\}|$. The h-index based structural diversity score of v, denoted by $h(v)$, is defined as $h(v) = \max\{r : \sum_r^n c_v(r) \geq r\}$. For short, diversity score is called.

Example 2 Consider the ego-network of v shown in Figure 1, subgraph H_1 is not a 2-core discriminative component since it can be further decomposed into two 3-cores H_4 and H_5. There is no discriminative core with the coreness of 1, so $c_v(1) = 0$. And $c_v(2) = 2$ since it has two discriminative cores H_2 and H_3 with the coreness of 2. Similarly, $c_v(3) = 2$ because H_4, H_5 are two discriminative cores with the coreness of 3. There exists no discriminative cores with coreness greater than 3. Thus, the diversity vector of v is $\mathcal{C}(v) = [0, 2, 2, 0, ..., 0]$. And the diversity score is $h(v) = 2$ by definition.

In this paper, we study the problem of h-index based structural diversity search in a graph. The problem formulation is defined as follows.

Problem formulation Given a graph G and an integer k, the goal of h-index based structural diversity search problem is to find an optimal answer S^* consisted of k vertices with the highest h-index based structural diversity scores, i.e.,

$$S^* = \underset{S \subseteq V, |S|=k}{\arg\max} \{\min_{v \in S} h(v)\}.$$

4 Baseline algorithm

In this section, we introduce a baseline approach for h-index based structural diversity search over graph G. The high-level idea is to compute the diversity score for each vertex in graph G one by one. After obtaining the scores of all vertices, it sorts vertices in decreasing order of their scores and returns the first k vertices with the highest structural

diversity scores. This method computes the top-k result from scratch, which is intuitive and straightforward to obtain answers.

In the following, we first introduce an existing algorithm of core decomposition [3]. Then, we present an important and useful procedure to compute h-index based structural diversity score $h(v)$ for a given vertex v.

4.1 Core decomposition

The core decomposition of graph G computes the coreness of all vertices $v \in V$. Algorithm 1 outlines the algorithm of core decomposition [3]. The algorithm starts with an integer $t = 1$, and iteratively removes the nodes with degree less than t and their incident edges. The number of $t - 1$ is assigned to be the coreness of the removed vertices. Then, the degree of affected vertices needs to be updated, since the removal of a vertex decreases the degree of its neighbors in the remaining graph. The number t is increased by one after each iteration, until all vertices and edges are deleted from the input graph.

4.2 Computing $h(v)$

The computation of $h(v)$ includes three major steps. First, we extract from graph G and obtain an ego-network $G_{N(v)}$ for vertex v, which is the induced subgraph of G by the set of v's neighbors $N(v)$. Next, we decompose the entire ego-network $G_{N(v)}$ into several discriminative cores, and count their corenesses to derive structural diversity vector $\mathcal{C}(v)$. The detailed procedure is outlined in Algorithm 2. Finally, based on the diversity vector of $\mathcal{C}(v)$, we compute the diversity score $h(v)$ by the Definition 5 using Algorithm 3.

Algorithm 1 Core decomposition [3].

Input: a graph $G = (V, E)$
Output: the coreness $\varphi_G(v)$ for each vertex $v \in V$

1: $\mathcal{L} \leftarrow$ Sort all vertices in G in ascending order of their degree.
2: Let $t \leftarrow 1$;
3: **while** G is not empty **do**
4: **for** each vertex $v \in \mathcal{L}$ with $d(v) < t$ **do**
5: Remove v and its incident edges from G; Remove v from $\mathcal{L}$;
6: $\varphi_G(v) \leftarrow t - 1$;
7: Update the degree of the affected vertices and reorder $\mathcal{L}$;
8: $t \leftarrow t + 1$;
9: **return** $\varphi_G(v)$ for each vertex $v \in V$;

Discriminative core decomposition Algorithm 2 outlines the detailed steps for discriminative core decomposition and diversity vector computation. For an ego-network $G_{N(v)}$ of vertex v, we firstly apply the core decomposition algorithm on it to calculate the coreness of each vertex (line 1). Then, we sort all vertices in $G_{N(v)}$ in ascending order of their coreness (line 3). For each integer t from 1 to the maximum coreness of the vertices in $G_{N(v)}$, we identify and count the number of discriminative cores with the coreness of t by using a breadth first search approach (lines 5–19). By definition, a discriminative core

with the coreness of t will be only formed by the vertices with the coreness of exactly t. Thus, in each iteration, we traverse vertices with the same coreness of t to search all the discriminative cores Hs with $\varphi(H) = t$ (lines 7–19 and lines 14–15). Edges connecting the current visited vertex x to the vertices with coreness greater than t indicate that the current found component can not be counted as a discriminative core and x does not belong to any discriminative cores in $G_{N(v)}$ (lines 16–17). Then the t-th element $c_v(t)$ of the diversity vector $\mathcal{C}(v)$ can be computed (lines 18–19). Finally, the diversity vector $\mathcal{C}(v)$ of v will be returned.

Algorithm 2 Discriminative core decomposition.

Input: an ego-network $G_{N(v)} = (N(v), \{(u, w) \in E : u, w \in N(v)\})$
Output: the diversity vector $\mathcal{C}(v)$

1: Apply the core decomposition algorithm in Algorithm 1 on $G_{N(v)}$;
2: $t_{max} = \max_{u \in N(v)} \varphi_{G_{N(v)}}(u)$;
3: $\mathcal{L} \leftarrow$ Sort all vertices in $G_{N(v)}$ in ascending order of their coreness;
4: $Q \leftarrow \emptyset$; $visited \leftarrow \emptyset$
5: **for** $t \leftarrow 1$ to t_{max} **do**
6: $\quad c_v(t) \leftarrow 0$;
7: $\quad$ **for** each vertex $u \in \mathcal{L}$ with the coreness of $\varphi_{G_{N(v)}}(u) = t$ **do**
8: $\quad\quad Flag \leftarrow$ **true**;
9: $\quad\quad$ **if** $u \notin visited$ **then**;
10: $\quad\quad\quad visited \leftarrow visited \cup \{u\}$; $Q.push(u)$;
11: $\quad\quad\quad$ **while** Q is not empty **do**
12: $\quad\quad\quad\quad x \leftarrow Q.pop()$;
13: $\quad\quad\quad\quad$ **for** each $y \in \{y : (x, y) \in E(G_{N(v)})\}$ **do**
14: $\quad\quad\quad\quad\quad$ **if** $\varphi_{G_{N(v)}}(y) = t$ **then**
15: $\quad\quad\quad\quad\quad\quad$ Insert y to Q and $visited$ if y is unvisited;
16: $\quad\quad\quad\quad\quad$ **else if** $\varphi_{G_{N(v)}}(y) > t$ **then**
17: $\quad\quad\quad\quad\quad\quad Flag \leftarrow$ **false**;
18: $\quad\quad\quad$ **if** $Flag =$ **true then**;
19: $\quad\quad\quad\quad c_v(t) \leftarrow c_v(t) + 1$;
20: **return** $\mathcal{C}(v)$;

Algorithm 3 Compute $h(v)$.

Input: a graph $G = (V, E)$; a vertex v
Output: the diversity score $h(v)$

1: Extract the ego-network $G_{N(v)}$ of v;
2: $\mathcal{C}(v) \leftarrow$ Apply the discriminative core decomposition procedure in Algorithm 2 on $G_{N(v)}$;
3: $h(v) \leftarrow 0$;
4: **for** $t \leftarrow t_{max}$ to 1 **do**
5: $\quad h(v) \leftarrow h(v) + c_v(t)$
6: $\quad$ **if** $h(v) \geq t$ **then** $h(v) \leftarrow t$; **break**;
7: **return** $h(v)$;

H-index score computation The details of computing the h-index based structural diversity score are shown in Algorithm 3. After figuring out the diversity vector $\mathcal{C}(v)$ (lines 1–2), the diversity score $h(v)$ can then be calculated by Definition 5 (lines 3–6). We firstly initialize $h(v)$ as 0 (line 3). Then, for each element $c_v(t)$ in the reverse order of the diversity vector $\mathcal{C}(v)$, we keep accumulating it to $h(v)$ until the first t appears such that $h(v) \geq t$ (line 4–6). Such t is the diversity score $h(v)$ of v.

Equipped with Algorithm 3, we are able to compute the h-index based structural diversity for all the vertices in G. By sorting the diversity scores, we can obtain the top-k results for a given k.

5 Efficient top-*k* search algorithm

The drawback of baseline method presented in the previous section is obviously inefficient and can be improved. Firstly, both the ego-network extraction and discriminative core decomposition are costly in computation. Secondly, it iteratively computes the h-index based structural diversity scores for all vertices on the entire graph G, which is expensive. Thirdly, some vertices appear to be obviously unqualified for the top-k result. And the score computations of them are reluctant and should be avoided.

In this section, we develop an efficient top-k search framework by exploiting useful pruning techniques to reduce the search space, leading to a small number of candidate vertices for score computations. Specifically, we design an upper bound $\widehat{h}(v)$ for diversity score $h(v)$, based on the analysis of the core structure.

5.1 An upper bound of $h(v)$

We starts with a structural property of t-core.

Lemma 1 *Given a vertex v and any vertex $u \in N(v)$, if u has $\varphi_{G_{N(v)}}(u) = r$ in* ego-network *$G_{N(v)}$, then u has the coreness $\varphi_G(u) \geq r + 1$ in graph G.*

Proof We omit the proof for brevity. The detailed proof can be referred to [21]. □

Example 3 Consider vertex x_1 in Figure 1, x_1 has coreness $\varphi_G(x_1) = 4$. However, in the ego-network $G_{N(v)}$, $\varphi_{G_{N(v)}}(x_1) = 3$. Here $\varphi_G(x_1) \geq \varphi_{G_{N(v)}}(x_1) + 1$ holds.

For a vertex v and some vertices $u \in N(v)$, the global coreness $\varphi_G(u)$ is sometimes much larger than the coreness of u in the ego-network of v, i.e. $\varphi_G(u) >> \varphi_{G_{N(v)}}(u)$. The following lemma gives another upper bound for estimating the coreness $\varphi_{G_{N(v)}}(u)$, w.r.t. vertices v and $u \in N(v)$.

Lemma 2 *Given a vertex v and its coreness $\varphi_G(v)$, $\forall u \in N(v)$, $\varphi_{G_{N(v)}}(u) < \varphi_G(v)$.*

Proof We prove this by contradiction. For any $u \in N(v)$, we assume $\varphi_G(v) = r$ and $\varphi_{G_{N(v)}}(u) \geq \varphi_G(v)$, which is $\varphi_{G_{N(v)}}(u) \geq r$. By the definition of coreness, there exists a subgraph $H \subseteq G_{N(v)}$ with coreness $\varphi(H) \geq r$ indicating that $\forall v^* \in V(H)$, $d_H(v^*) \geq r$.

We add the vertex v and its incident edges to H to generate a new subgraph $H' \subseteq G$, where $V(H') = V(H) \cup \{v\}$ and $E(H') = E(H) \cup \{(v,u) : u \in V(H)\}$. It's easy to verify that for all v^* in H', we have $d_{H'}(v^*) \geq r+1$. Since v is also contained in H', by definition, $\varphi_G(v) \geq r+1$, which contradicts to the condition $\varphi_G(v) = r$. □

Combining Lemmas 1 and 2, we have the following corollary.

Corollary 1 *Given a vertex v in graph G, for any vertex $u \in N(v)$, $\widehat{\varphi}_{G_{N(v)}}(u) = \min\{\varphi_G(v), \varphi_G(u) - 1\}$ and $\widehat{\varphi}_{G_{N(v)}}(u) \geq \varphi_{G_{N(v)}}(u)$ hold.*

Based on Corollary 1, we derive an upper bound $\widehat{h}(v)$ for the h-index based structural diversity score $h(v)$ as follows.

Lemma 3 *Given a vertex v and its* ego-network $G_{N(v)}$*, we have an upper bound of diversity score $h(v)$, denoted by*

$$\widehat{h}(v) = \max_{x \in \mathbb{Z}_+} \{x : |\{u \in N(v) : \widehat{\varphi}_{G_{N(v)}}(u) \geq x\}| \geq x \cdot (x+1)\}.$$

Proof Assume that $h(v) = x^*$, we prove $\widehat{h}(v) \geq x^*$. By $h(v) = x^*$, it indicates that there exist x^* discriminative cores g with $\varphi(g) \geq x^*$ in the ego-network $G_{N(v)}$. For $\varphi(g) \geq x^*$, discriminative core g has at least $x^* + 1$ nodes u with $\varphi_{G_{N(v)}}(u) \geq x^*$. Thus, the whole ego-network $G_{N(v)}$ has at least $x^* \cdot (x^* + 1)$ nodes u with $\varphi_{G_{N(v)}}(u) \geq x^*$, i.e., $h(v) = x^* \leq \max_{x \in \mathbb{Z}_+}\{x : |\{u \in N(v) : \varphi_{G_{N(v)}}(u) \geq x\}| \geq x \cdot (x+1)\}$. By Corollary 1, $\widehat{\varphi}_{G_{N(v)}}(u) \geq \varphi_{G_{N(v)}}(u)$, hence we have $\widehat{h}(v) \geq x^* = h(v)$. □

According to Lemma 3, once applying the core decomposition algorithm on graph G, we can directly compute the upper bounds $\widehat{h}(v)$ for all vertices v.

5.2 Top-K structural diversity search framework

Equipped with the upper bound $\widehat{h}(v)$, we develop an efficient top-k search framework for safely pruning the search space and avoiding the unnecessary computation of $h(v)$. The efficient top-k structural diversity search framework is presented in Algorithm 4.

Algorithm 4 starts with the initialization of the upper bound of each vertex v (lines 1–2). Then, it sorts all vertices in descending order according to their upper bounds (line 3). It maintains a list $\mathcal{S}$ to store the top-k result (line 4). In each iteration, the algorithm pops out a vertex v^* from the vertex list $\mathcal{L}$ with the largest upper bound $\widehat{h}(v^*)$ (line 6). Next, it checks the early stop condition: if the answer set $\mathcal{S}$ has k results and the minimum score in $\mathcal{S}$ is no less than the current upper bound, i.e. $\widehat{h}(v^*) \leq min_{v \in \mathcal{S}} h(v)$, the current vertex v^* is safely pruned and the searching process is terminated (lines 8-9). Otherwise, the procedure of structural diversity score computation is invoked and check if v^* can be added into the result set (lines 10-14). Finally, the top-k results stored in $\mathcal{S}$ are returned.

Algorithm 4 Efficient top-k search framework.

Input: $G = (V, E)$, an integer k
Output: top-k structural diversity results

1: Apply the core decomposition on G by Algorithm 1 and obtain $\varphi_G(v)$ for all vertices $v \in V$;
2: **for** $v \in V$ **do**
3: Compute $\widehat{h}(v)$ according to Lemma 3;
4: $\mathcal{L} \leftarrow$ Sort all vertices V in descending order of $\widehat{h}(v)$;
5: $\mathcal{S} \leftarrow \emptyset$;
6: **while** $\mathcal{L} \neq \emptyset$ **do**
7: $v^* \leftarrow \arg\max_{v \in \mathcal{L}} \widehat{h}(v)$; Delete v^* from $\mathcal{L}$;
8: **if** $|\mathcal{S}| = k$ and $\widehat{h}(v^*) \leq \min_{v \in \mathcal{S}} h(v)$ **then**
9: **break**;
10: Invoke Algorithm 3 to compute $h(v^*)$;
11: **if** $|\mathcal{S}| < k$ **then** $\mathcal{S} \leftarrow \mathcal{S} \cup \{v^*\}$;
12: **else if** $h(v^*) > \min_{v \in \mathcal{S}} h(v)$ **then**
13: $u \leftarrow \arg\min_{v \in \mathcal{S}} h(v)$;
14: $\mathcal{S} \leftarrow (\mathcal{S} - \{u\}) \cup \{v^*\}$;
15: **return** $\mathcal{S}$;

5.3 Complexity analysis

In this section, we analyze the time and space complexity of Algorithm 4.

Lemma 4 *Algorithm 3 computes $h(v)$ for each vertex v in $O(\sum_{u \in N(v)} min\{d(u), d(v)\})$ time and $O(m)$ space.*

Proof Extracting $G_{N(v)}$ of v takes $O(\sum_{u \in N(v)} min\{d(u), d(v)\})$, since all triangles $\triangle_{vuw}$ should be listed to enumerate each edge $(u, w) \in E(G_{N(v)})$. According to [3], the core decomposition performed in $G_{N(v)}$ takes $O(|E(G_{N(v)})| + d(v))$ time. The sorting of the vertices can be finished in $O(d(v))$ time using bin sort. And the breadth first search process for identifying the discriminative cores needs $O(|E(G_{N(v)})|)$ time. In addition, the computing of the h-index based structural diversity score $h(v)$ runs in $O(\delta(G_{N(v)}))$ time, where $\delta(G_{N(v)}) = \max_{u \in N(v)} \varphi_{G_{N(v)}}(u)$ is the degeneracy of $G_{N(v)}$. And $\delta(G_{N(v)})$ is bounded by the degree of v, which is $O(\delta(G_{N(v)})) \subseteq d(v)$. Overall, the time complexity of Algorithm 3 is $O(\sum_{u \in N(v)} min\{d(u), d(v)\})$.

We continue to analyze the space complexity of Algorithm 3. The storage of the ego-network of v takes $O(n + m)$ space since $G_{N(v)} \subseteq G$. And both the sorted list of vertices (line 4) and the structural diversity vector of v takes $O(n)$ space. Thus, the space complexity of Algorithm 3 is $O(n + m) \subseteq O(m)$ due to our graph connectivity assumption. □

Theorem 1 *Algorithm 4 computes the top-k results in $O(\rho m)$ time and $O(m)$ space, where ρ is the arboricity of G. According to* [9]*, $\rho \leq \min\{d_{max}, \sqrt{m}\}$.*

Proof Firstly, the core decomposition algorithm performed on G takes $O(m)$ time and $O(n + m)$ space. Secondly, the computation of upper bound $\widehat{h}(v)$ for all v's takes $O(m)$

time and $O(n)space$. In the worst case, Algorithm 4 needs to compute $h(v)$ for every vertex v. This takes $O(\sum_{v\in V}\{\sum_{u\in N(v)} min\{d(u), d(v)\}\})$ time in total by Lemma 4. According to [9], we have

$$O\left(\sum_{v\in V}\left\{\sum_{u\in N(v)} min\{d(u), d(v)\}\right\}\right) \subseteq O\left(\sum_{(u,v)\in E} min\{d(u), d(v)\}\right) \subseteq O(\rho m).$$

Here ρ is the arboricity of graph G, which is defined as the minimum number of disjoint spanning forests that cover all the edges in G. In addition, the top-k results can be maintained in a list in $O(n)$ time and $O(n)$ space using bin sort. Overall, Algorithm 4 runs in $O(\rho m)$ time and $O(m)$ space. □

6 Parallel top-*k* search algorithm

In this section, we present a parallel algorithm for top-k structural diversity search in a multiprocessor shared memory machine.

6.1 Parallelization analysis

We first briefly give an overview of the new parallel top-k search algorithm.

Overview We analyze the diversity score computation in Algorithm 3. We observe that the algorithm of computing score $h(v)$ is conducted in its own ego-network $G_{N(v)}$. Thus, the diversity score $h(v)$ computation for different vertices $v \in V$ can be treated independently and operated using multiple threads in parallel. This observation allows us to apply parallel computing for speeding up the top-k structural diversity search. An intuitive method is to compute the diversity scores for all vertices in multi-threading parallelization. However, this method straightforwardly enumerates all vertices without any pruning strategies, which may be still inefficient for large-scale graphs. To further improve the efficiency, we focus on developing a parallel algorithm based on the efficient top-k structural diversity search framework in Algorithm 4.

In the following, we identify key steps in the top-k search framework in Algorithm 4 and analyze their synchronicity/parallelization. Generally, the top-k search framework has four key steps: upper bound estimation (lines 3–4 of Algorithm 4), score computation (line 10 of Algorithm 4), pruning condition validation (lines 8-9 of Algorithm 4), and top-k answer update (lines 11–14 of Algorithm 4).

Parallel computation in estimating upper bounds and ranking vertices In Algorithm 4, two important initialization steps are computing the upper bounds for all vertices and ranking vertices in decreasing order of their upper bounds. We consider the parallelization of computing upper bound and vertex ranking as follows. According to the definition of upper bound in Lemma 3, the upper bound of $\widehat{h}(v)$ for a given vertex v is determined by its ego-network $G_{N(v)}$. Hence, the upper bound computations for different vertices are independent, which allow us to process them in parallel. On the other hand, Algorithm 4 ranks vertices using the bin sort, which inserts a vertex v into a corresponding bin with the value of its upper bound $\widehat{h}(v)$. However, this step needs to be operated synchronously. Because the information loss may occur when two threads deal with the same bin simultaneously. Thus, we compute the upper bounds asynchronously and rank vertices synchronously.

Parallel search order and pruning strategy We develop a new search order and pruning condition for parallel top-k search. Assume that we insist on randomly assigning vertices for score computation in multiple threads and directly adopting the early stop condition in Algorithm 4. This strategy leads to inaccurate answers. This is because that the whole algorithm may directly terminate for an existing vertex v^* with $\widehat{h}(v^*) \leq min_{v \in \mathcal{S}} h(v)$ in a thread, thus it may dismiss the score computations of vertices in other threads, which should be the answers.

To ensure the correctness of parallel algorithm, we define a parallel order of score computations. Let $B[i]$ be the set of vertices whose upper bounds are i, i.e., $B[i] = \{v \in V : \widehat{h}(v) = i\}$. The parallel algorithm computes the structural diversity score for vertices $B[i]$ in the decreasing order of upper bound i. If and only if the scores of all vertices in $B[i]$ have been calculated, we next consider the score computation of vertices in $B[i-1]$. Thus, no vertex with a smaller upper bound is computed in advanced in the multi-threads schema, which sacrifices the efficiency but ensures the answer exactness.

Lemma 5 *Given an integer $i \geq 0$, all vertices $v \in B[j]$ where $0 \leq j \leq i$, can be safely pruned if and only if the following conditions are fulfilled at the same time:*

- *The size of top-k answer $\mathcal{S}$ is k, i.e., $|\mathcal{S}| = k$.*
- *For each vertex $v \in B[l]$ where $l > i$, the score $h(v)$ has been computed.*
- *The smallest score in the answer S is no less than i, i.e., $i \leq \min_{v \in \mathcal{S}} h(v)$.*

Proof We first separate the vertices into two independent sets $\mathcal{X}$ and $\mathcal{Y}$, where $\mathcal{X} \cup \mathcal{Y} = V$ and $\mathcal{X} \cap \mathcal{Y} = \emptyset$. Let $\mathcal{X} = \bigcup_{l=i+1}^{maxUB} B[l]$ be the set of vertices with upper bounds greater than i, where $maxUB$ is the maximum upper bound of all vertices. Let $\mathcal{Y} = \bigcup_{j=minUB}^{i} B[j]$ be the set of vertices whose upper bounds are smaller or equal to i, where $minUB$ is the minimum upper bound of all vertices.

In the first place, we show that under the above conditions, $\mathcal{S}$ will be the exact top-k results among all vertices whose upper bounds are greater than i, i.e., $\forall v \in \mathcal{X}$. This is quite straightforward. By the second condition, the diversity scores of vertices in sets $B[l]$'s for all $l > i$ are completely computed, which implies that the the score of each of them is compared and updated in $\mathcal{S}$. And $\mathcal{S}$ always keeps the largest values of scores of the vertices in $\mathcal{X}$. Moreover, combining with the first condition that the size of $\mathcal{S}$ is k, we can conclude that in the current iteration, $\mathcal{S}$ is the top-k results among the vertices in $\mathcal{X}$, i.e., $\forall v^* \in \mathcal{X} - \mathcal{S}, h(v^*) \leq \min_{v \in \mathcal{S}} h(v)$ and $|\mathcal{S}| = k$.

In the second place, we further show that $\mathcal{S}$ is the top-k results among all vertices. We prove this by contradiction. Assume that under the above three conditions, the algorithm stops with inaccurate results. Specifically, we assume that there exists a vertex u with structural diversity score $h(u)$ larger than the smallest diversity score in the top-k list $\mathcal{S}$, i.e., $h(u) > \min_{v \in \mathcal{S}} h(v)$. And u has upper bound smaller than or equal to i, i.e., $u \in \mathcal{Y}$ and $\widehat{h}(u) \leq i$. By our assumption, we have $i \geq \widehat{h}(u) \geq h(u) > \min_{v \in \mathcal{S}} h(v)$, which is $i > \min_{v \in \mathcal{S}} h(v)$. This contradicts with the third condition that $i \leq \min_{v \in \mathcal{S}} h(v)$. Thus, $\mathcal{S}$ is the exact top-k results of all vertices. And the rest computations can be pruned safely. □

Equipped with this new pruning strategy in Lemma 5, the parallel algorithm can compute diversity scores for all vertices in a set of $B[i]$ asynchronously, one by one in decreasing order of i. However, the scores of all vertices in the same $B[i]$ can be computed simultaneously in parallel. This strategy achieves the parallel efficiency and ensures the answer correctness.

Synchronous top-k answer update We update the top-k answer S synchronously. Specifically, only one thread can handle the update of top-k answer S, deciding a vertex u whether should be added into the answer set S. This step is not computed in parallel, due to the possible conflicts may happen among different threads for updating S simultaneously.

6.2 Parallel top-k structural diversity search

Algorithm 5 presents the details of parallel top-k structural diversity search algorithm. The algorithm firstly computes the upper bounds of all vertices in parallel (lines 3–4), and applies the bin sort on vertices in decreasing order of their upper bounds synchronously (lines 5–7). It computes the maximum and minimum values of upper bounds denoted as $maxUB$ and $minUB$ (line 8). Next, it sequentially traverses the set $B[i]$ one by one in decreasing order of i where $minUB \leq i \leq maxUB$ (line 9). In each iteration of traversing $B[i]$, it computes scores for all vertices in $B[i]$ in parallel, and also synchronously checks the early stop condition and update the answer (lines 9–20). Specifically, it checks an early stop condition by Lemma 5: if $|\mathcal{S}| = k$ and the smallest diversity score in $\mathcal{S}$ is no less than i where i represents the largest value of upper bounds of the remaining vertices in our parallel setting (lines 11–12), the search process is early terminated. Otherwise, it computes score for all vertices in $B[i]$ in parallel (lines 13–14) and updates the top-k list $\mathcal{S}$ synchronously to ensure the answer correctness (line 15–20).

Finally, it returns the top-k results (line 21).

Algorithm 5 Parallel top-k structural diversity search.

Input: $G = (V, E)$, an integer k, a number of threads T
Output: top-k structural diversity results

1: Apply the core decomposition on G by Algorithm 1 and obtain $\varphi_G(v)$ for all vertices $v \in V$;
2: $B \leftarrow \emptyset$; $\mathcal{S} \leftarrow \emptyset$;
3: **parallel for** $v \in V$ **do** // Compute upper bounds for all vertices in parallel.
4: Compute $\widehat{h}(v)$ according to Lemma 3;
5: **synchronize begin** // Synchronously assign the vertices to their corresponding set.
6: $B[\widehat{h}(v)] \leftarrow B[\widehat{h}(v)] \cup \{v\}$;
7: **synchronize end**
8: $minUB \leftarrow \min_{v \in V} \widehat{h}(v)$; $maxUB \leftarrow \max_{v \in V} \widehat{h}(v)$;
9: **for** $i \leftarrow maxUB$ to $minUB$ **do** // Synchronously process each set $B[i]$ in decreasing order of i.
10: **if** $B[i] = \emptyset$ **then continue**; // Pruning $B[i]$ as an empty set.
11: **if** $|\mathcal{S}| = k$ and $i \leq \min_{v \in \mathcal{S}} h(v)$ **then** // Check for the early stop conditions.
12: **break**;
13: **parallel for** each $v^* \in B[i]$ **do** // Process vertices in the same bin in parallel.
14: Invoke Algorithm 3 to compute $h(v^*)$;
15: **synchronize begin** // Synchronously update the top-k answer $\mathcal{S}$.
16: **if** $|\mathcal{S}| < k$ **then** $\mathcal{S} \leftarrow \mathcal{S} \cup \{v^*\}$;
17: **else if** $h(v^*) > \min_{v \in \mathcal{S}} h(v)$ **then**
18: $u \leftarrow \arg\min_{v \in \mathcal{S}} h(v)$;
19: $\mathcal{S} \leftarrow (\mathcal{S} - \{u\}) \cup \{v^*\}$;
20: **synchronize end**
21: **return** $\mathcal{S}$;

Table 1 Network statistics

Name	$\|V\|$	$\|E\|$	d_{max}
Gowalla	196,591	950,327	14,730
Youtube	1,134,890	2,987,624	28,754
LiveJournal	3,997,962	34,681,189	14,815
Orkut	3,072,441	117,185,083	33,313

7 Experiments

We conduct extensive experiments on real-world datasets to evaluate the effectiveness and efficiency of our proposed h-index based structural diversity model and algorithms.

Datasets We run our experiments on four real-world datasets downloaded on the SNAP website [32]. All datasets are treated as undirected graphs. The statistics of the networks are listed in Table 1. We report the node size $|V|$, edge size $|E|$ and the maximum degree d_{max} of each network.

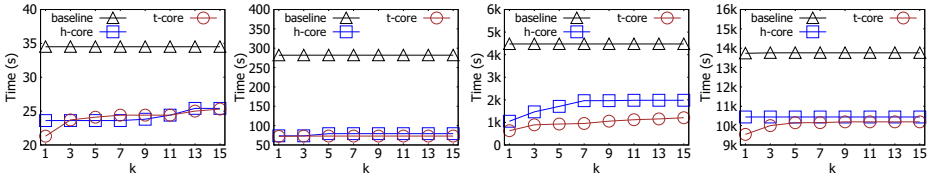
Compared methods We evaluate all compared methods in terms of efficiency, effectiveness and also sensitivity to parameter setting. Specifically, we show four compared algorithms as follows.

- baseline: is the baseline method proposed in Section 4.
- h-core: is an improved top-k search algorithm for computing the top-k vertices with highest h-index based structural diversity in Algorithm 4.
- t-core: is to compute the top-k vertices with highest t-core based structural diversity [21]. Here, t is a parameter of coreness threshold.
- h-core-P: is a parallel algorithm for top-k structural diversity search using T threads in Algorithm 5. Here, we set $1 \leq T \leq 64$ in experiments.

Note that in the sensitivity evaluation, we test the state-of-the-art competitor t-core and compare the top-k results for different parameter t. Our h-index based structural diversity model has no input parameter, which is consistent on the top-k results.

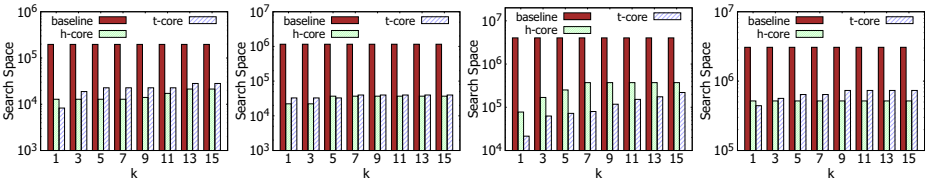
7.1 Efficiency evaluation

In the first experiment, we compare the efficiency of baseline, h-core and t-core on four real-world datasets. For the t-core method, we fix parameter $t = 2$. We compare the running time and search space (i.e., the number of vertices whose structural diversity scores are computed in the search process). Figure 2 shows the running time results of three methods varied by



(a) Gowalla (b) Youtube (c) LiveJournal (d) Orkut

Figure 2 Comparsion of baseline, h-core and t-core in terms of running time (in seconds) itemize



(a) Gowalla (b) Youtube (c) LiveJournal (d) Orkut

Figure 3 Comparsion of baseline, h-core and t-core in terms of search space

k. It clearly shows that top-k search algorithm h-core runs much faster than baseline on all the reported datasets. Specifically, in Figure 2c, h-core is 5 times faster than baseline on Youtube in term of running time. Moreover, Figure 3 further shows the search space of three methods varied on all datasets. We can observe that leveraging on the upper bound $\widehat{h}(v)$, a large number of disqualified vertices is pruned during the search process by h-core. The search space significantly shrinks into less than $\frac{1}{10}$ of vertex size in graphs. It verifies the tightness of our upper bound and the superiority of h-core against baseline in efficiency. According to Figures 2 and 3, our h-core is very comparative to the state-of-art method t-core in terms of running time and search space.

In the second experiment, we compare the efficiency of h-core and h-core-P on four large real-world graph datasets. To enable multi-thread programming with Intel CPU, we use the well-known C++ parallelization computing library OpenMP.[1] We vary the number of threads T in $\{1, 2, 4, 8, 16, 32, 64\}$ and report the running time in seconds. Figure 4 shows the running time results of two methods by varing different thread numbers. As we can see, as the thread numbers increased, the efficiency of h-core-P is significantly improved by the parallelization scheme in h-core-P. Especially, on the largest dataset of Orkut, h-core-P using 64 threads achieves 13X speedup compared with the running time of h-core.

7.2 Sensitivity evaluation

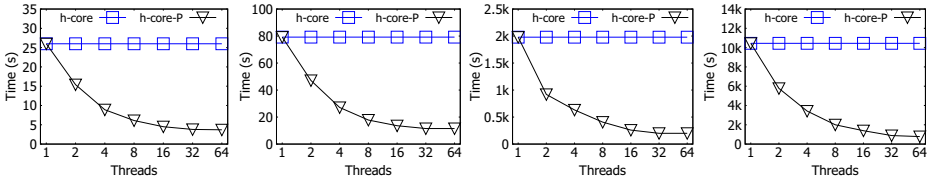
This experiment evaluates the sensitivity of t-core model. Given two different values of t, t-core model may generate two different lists of top-k ranking results. We use the Kendall rank tau distance to counts the number of pairwise disagreements between two top-k lists. The larger the distance, the more dissimilar the two lists, and also more sensitive the t-core model. We adopt the Kendall distance with penalty, denoted by,

$$K^{(p)}(\tau_1, \tau_2) = \sum_{\{i,j\} \in \mathcal{P}} \overline{K}_{i,j}^{(p)}(\tau_1, \tau_2)$$

where $\mathcal{P}$ is the set of all unordered pairs of distinct elements in two top-k list τ_1, τ_2 and p is the penalty parameter. In our setting, we set $p = 1$ and normalize the Kendall distance by the number of permutation $|\mathcal{P}|$. The values of normalized Kendall distance range from 0 to 1.

We test the sensitivity of t-core model by varying parameter t in $\{2, 4, 6, 8, 10\}$. We compute the Kendall distance of two top-100 lists by t-core model with two different t. The results of sensitivity heat matrix on four datasets are shown in Figure 5. The darker colors reveal larger Kendall distances between two top-k lists and also more sensitive of t-core

[1] https://www.openmp.org/



(a) Gowalla (b) Youtube (c) LiveJournal (d) Orkut

Figure 4 Comparsion of h-core and h-core-P in terms of running time (in seconds)

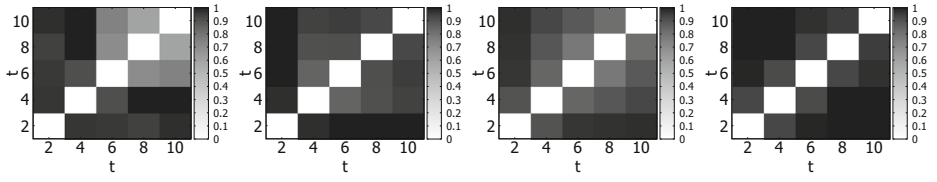
models on this pair of parameters t. Overall, sensitivity heat matrices are depicted in dark for most parameter settings on all datasets. This reflects that the top-k results computed by t-core are very sensitive to the setting of parameter t, which has a bad robustness. It strongly indicates the necessity and importance of our parameter-free structural diversity model.

7.3 Effectiveness evaluation

In this experiment, we evaluate the effectiveness of our proposed h-index based structural diversity. We compare our method h-core with state-of-the-art t-core [21] in the task of social contagion. Specifically, we adopt the independent cascade model to simulate the influence propagation process in graphs [17]. Influential probability of each edge is set to 0.01. Then, we select 50 vertices as activated seeds by an influence maximization algorithm [46]. We perform 1000 times of Monte Carlos sampling for propagation. For comparison, we count the number of activated vertices in the top-k results by t-core and h-core methods. The method that achieves the largest number of activated vertices is regarded as the winner.

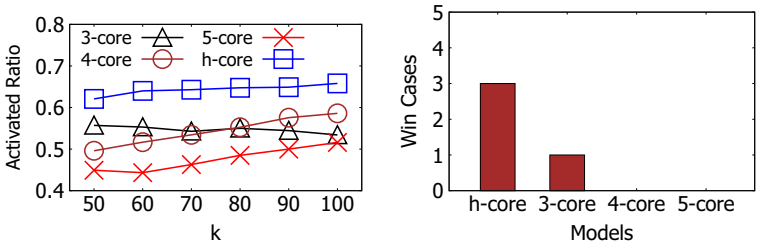
First, we report the average activated rate by h-core and t-core method on all four datasets in Figure 6a. Let D ={"Gowalla","Youtube", "LiveJournal","Orkut"}. Given a dataset $d \in D$, the activated rate is defined as $f_k(d) = \frac{ActNum_k}{k}$, where $ActNum_k$ is the number of activated vertices in the top-k result. The average activated rate is defined as $ActRate_k = \frac{\sum_{d \in D} f_k(d)}{|D|}$. Figure 6a shows that our method h-core achieves the highest activated rates, which significantly outperforms t-core method for all different t. It indicates that the top-k results found by h-core tend to have higher probability to be affected in social contagion.

In addition, we also report the winning cases of h-core and t-core with different parameter t on all dataset. We vary $t = \{2, 3, 4\}$ and set $k = 100$ for all methods. The winner of a dataset is the method that achieves the highest number of activated vertices in this dataset. Figure 6b shows the winning cases of t-core and h-core. As we can see, h-core wins on three datasets, which achieves the best performance. It further shows the superiority of our



(a) Gowalla (b) Youtube (c) LiveJournal (d) Orkut

Figure 5 Sensitivity heat matrices of t-core model on all datasets. Each matrix element represents the Kendall's Tau distance between two top-100 ranking lists by t-core model with different t



(a) Average Activated Rate (b) Winning Cases Distribution

Figure 6 Comparison of t-core and h-core in terms of the average activated ratio and winning cases on four datasets

h-index structural diversity model. Besides, 3-core wins once, 4-core and 5-core win none, indicating that t-core performs sensitively to parameter t.

8 Conclusion and future work

In this paper, we propose a parameter-free structural diversity model based on h-index and study the top-k structural diversity search problem. To solve the top-k structural diversity search problem, an upper bound for the diversity score and a top-k search framework for efficiently reducing the search space are proposed. To accelerate the computation, we design a novel parallelization scheme for efficient top-k structural diversity search. Extensive experiments on real-wold datasets verify the efficiency of our pruning techniques and the effectiveness of our proposed h-index based structural diversity model.

In the future work, we would like to improve the current online top-k search approach. We can adopt a graph indexing method to keep important structural information to accelerate the top-k query processing. Next, we only deal with undirected static graphs instead of other types of graphs such as dynamic graphs, directed graphs, attributed graphs, and so on. Defining social contexts and finding parameter-free structural diversity over other kinds of social networks (e.g., public-private social networks [28]) are also wide open.

Acknowledgments This work is supported by the NSFC Nos. 61702435, 61972291, RGC Nos. 12200917, 12200817, CRF C6030-18GF, and the National Science Foundation of Hubei Province No. 2018CFB519.

References

1. Aridhi, S., Brugnara, M., Montresor, A., Velegrakis, Y.: Distributed k-core decomposition and maintenance in large dynamic graphs. In: DEBS, pp. 161–168. ACM (2016)
2. Arifuzzaman, S., Khan, M., Marathe, M.: Fast parallel algorithms for counting and listing triangles in big graphs. TKDD **14**(1), 1–34 (2019)
3. Batagelj, V., Zaversnik, M.: An o (m) algorithm for cores decomposition of networks arXiv preprint cs/0310049 (2003)
4. Bonchi, F., Gullo, F., Kaltenbrunner, A., Volkovich, Y.: Core decomposition of uncertain graphs. In: KDD, pp. 1316–1325. ACM (2014)
5. Chang, L., Qin, L.: Cohesive subgraph computation over large sparse graphs: algorithms, data structures, and programming techniques. Springer, Berlin (2018)
6. Chang, L., Zhang, C., Lin, X., Qin, L.: Scalable top-k structural diversity search. In: ICDE, pp. 95–98 (2017)

7. Cheng, H., Zhong, M., Wang, J., Qian, T.: Keyword search based mashup construction with guaranteed diversity. In: DEXA, pp. 423–433 (2019)
8. Cheng, J., Ke, Y., Chu, S., Özsu, M.T.: Efficient core decomposition in massive networks. In: ICDE, pp. 51–62 (2011)
9. Chiba, N., Nishizeki, T.: Arboricity and subgraph listing algorithms. SIAM J. Comput. **14**(1), 210–223 (1985)
10. Ding, F., Zhuang, Y.: Ego-network probabilistic graphical model for discovering on-line communities. Appl Intell. **48**(9), 3038–3052 (2018)
11. Esfandiari, H., Lattanzi, S., Mirrokni, V.: Parallel and streaming algorithms for k-core decomposition. In: ICML, pp. 1397–1406 (2018)
12. Fang, Y., Cheng, R., Luo, S., Hu, J.: Effective community search for large attributed graphs. Proc. VLDB Endow. **9**(12), 1233–1244 (2016)
13. Fang, Y., Wang, Z., Cheng, R., Wang, H., Hu, J.: Effective and efficient community search over large directed graphs. IEEE Trans. Knowl. Data Eng. **31**(11), 2093–2107 (2018)
14. Fang, Y., Huang, X., Qin, L., Zhang, Y., Zhang, W., Cheng, R., Lin, X.: A survey of community search over big graphs 1–40. VLDB J **29**(1), 353–392 (2020)
15. Fang, Y., Yang, Y., Zhang, W., Lin, X., Cao, X.: Effective and efficient community search over large heterogeneous information networks . Proc VLDB Endow **13**(6), 854–867 (2020)
16. Galimberti, E., Bonchi, F., Gullo, F.: Core decomposition and densest subgraph in multilayer networks. In: CIKM, pp. 1807–1816. ACM (2017)
17. Goyal, A., Lu, W., Lakshmanan, L.V.S.: CELF++: optimizing the greedy algorithm for influence maximization in social networks. In: WWW, pp. 47–48 (2011)
18. Hirsch, J.E.: An index to quantify an individual's scientific research output. Proc. Natl. Acad. Sci. **102**(46), 16569–16572 (2005)
19. Hou, J., Wang, S., Wu, G., Fu, G., Jia, S., Wang, Y., Li, B., Zhang, L.: Parallel strongly connected components detection with multi-partition on gpus. In: ICCS, pp. 16–30. Springer (2019)
20. Huang, X., Cheng, H., Li, R.-H., Qin, L., Yu, J.X.: Top-k structural diversity search in large networks. PVLDB **6**(13), 1618–1629 (2013)
21. Huang, X., Cheng, H., Li, R., Qin, L., Yu, J.X.: Top-k structural diversity search in large networks. VLDB J. **24**(3), 319–343 (2015)
22. Huang, X., Cheng, H., Yu, J.X.: Attributed community analysis: global and ego-centric views. IEEE Data Eng. Bull. **39**(3), 29–40 (2016)
23. Huang, J., Huang, X., Zhu, Y., Xu, J.: Parameter-free structural diversity search. In: WISE, pp. 677–693 (2019)
24. Huang, X., Lakshmanan, L.V., Xu, J.: Community search over big graphs. Morgan & Claypool Publishers, San Rafael (2019)
25. Huang, J., Huang, X., Xu, J.: Truss-based structural diversity search in large graphs. arXiv preprint arXiv:2007.05437 (2020)
26. Huckfeldt, R.R., Sprague, J.: Citizens, Politics and Social Communication: Information and Influence in an Election Campaign. Cambridge University Press, Cambridge (1995)
27. Jakma, P., Orczyk, M., Perkins, C.S., Fayed, M.: Distributed k-core decomposition of dynamic graphs. In: StudentWorkshop@CoNEXT, pp. 39–40. ACM (2012)
28. Jiang, J., Huang, X., Choi, B., Xu, J., Bhowmick, S.S., ppkws, L.X.u.: An efficient framework for keyword search on public-private networks. In: ICDE, pp. 457–468 (2020)
29. Jin, J., Luo, J., Khemmarat, S., Dong, F., Gao, L.: Gstar: an efficient framework for answering top-k star queries on billion-node knowledge graphs. World Wide Web **22**(4), 1611–1638 (2019)
30. Kempe, D., Kleinberg, J.M., Tardos, É.: Maximizing the spread of influence through a social network. In: KDD, pp. 137–146 (2003)
31. Kim, M.-S., Lee, S., Han, W.-S., Park, H., Lee, J.-H.: Dsp-cc-: I/o efficient parallel computation of connected components in billion-scale networks. ICDE **27**(10), 2658–2671 (2015)
32. Leskovec, J., Krevl, A.: SNAP Datasets: Stanford large network dataset collection http://snap.stanford.edu/data (2014)
33. Levorato, V.: Core decomposition in directed networks: Kernelization and strong connectivity. In: CompleNet, vol. 549, pp. 129–140 (2014)
34. Li, R., Yu, J.X., Mao, R.: Efficient core maintenance in large dynamic graphs. TKDE **26**(10), 2453–2465 (2014)
35. Liu, G., Shi, Q., Zheng, K., Li, Z., Liu, A., Xu, J.: Context-aware graph pattern based top-k designated nodes finding in social graphs. World Wide Web **22**(2), 751–770 (2019)
36. Liu, Q., Zhu, Y., Zhao, M., Huang, X., Xu, J., Gao, Y.: Vac: vertex-centric attributed community search. In: ICDE, pp. 937–948 (2020)

37. Mcauley, J., Leskovec, J.: Discovering social circles in ego networks. TKDD **8**(1), 4 (2014)
38. Montresor, A., Pellegrini, F.D., Miorandi, D.: Distributed k-core decomposition. TPDS **24**(2), 288–300 (2013)
39. Nguyen, D., Lenharth, A., Pingali, K.: A lightweight infrastructure for graph analytics. In: SOSP, pp. 456–471 (2013)
40. Sanz-Cruzado, J., Pepa, S.M., Castells, P.: Structural novelty and diversity in link prediction. In: Companion Proceedings of the The Web Conference, vol. 2018, pp. 1347–1351 (2018)
41. Sarıyüce, A.E., Gedik, B., Jacques-Silva, G., Wu, K.-L., Çatalyürek, Ü.V.: Streaming algorithms for k-core decomposition. PVLDB **6**(6), 433–444 (2013)
42. Shang, Z.J., Yu, X., Zhang, Z.: Tufast: a lightweight parallelization library for graph analytics. In: ICDE, pp. 710–721. IEEE (2019)
43. Shun, J., Blelloch, G.E.: Ligra: a lightweight graph processing framework for shared memory. In: Proceedings of the 18th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pp. 135–146 (2013)
44. Smith, S., Liu, X., Ahmed, N.K., Tom, A.S., Petrini, F., Karypis, G.: Truss decomposition on shared-memory parallel systems. In: HPEC, pp. 1–6. IEEE (2017)
45. Su, J., Kamath, K., Sharma, A., Ugander, J., Goel, S.: An experimental study of structural diversity in social networks. arXiv preprint arXiv:1909.03543 (2019)
46. Tang, Y., Shi, Y., Xiao, X.: Influence maximization in near-linear time: a martingale approach. In: SIGMOD Conference, pp. 1539–1554. ACM (2015)
47. Ugander, J., Backstrom, L., Marlow, C., Kleinberg, J.: Structural diversity in social contagion. PNAS **109**(16), 5962–5966 (2012)
48. Wang, W., Gu, Y., Wang, Z., Yu, G.: Parallel triangle counting over large graphs. In: DASFAA, pp. 301–308. Springer (2013)
49. Wang, R., Wang, S., Zhou, X.: Parallelizing approximate single-source personalized pagerank queries on shared memory. VLDBJ **28**(6), 923–940 (2019)
50. Wen, D., Qin, L., Zhang, Y., Lin, X., Yu, J.X.: I/O efficient core graph decomposition Application to degeneracy ordering. TKDE **31**(1), 75–90 (2019)
51. Wu, H., Cheng, J., Lu, Y., Ke, Y., Huang, Y., Yan, D., Wu, H.: Core decomposition in large temporal graphs. In: BigData, pp. 649–658 (2015)
52. Zhang, Y., Yu, J.X., Zhang, Y., Qin, L.: A fast order-based approach for core maintenance. In: ICDE, pp. 337–348 (2017)
53. Zhang, Y., Wang, L., Zhu, J.J., Wang, X., Pentland, A.: The strength of structural diversity in online social networks. arXiv preprint arXiv:1906.00756 (2019)
54. Zhang, Q., Li, R., Yang, Q., Wang, G., Qin, L.: Efficient top-k edge structural diversity search. In: ICDE, pp. 205–216 (2020)