Efficient Core Propagation based Hierarchical Graph Clustering

Jinbin Huang
Hong Kong Baptist University
jbhuang@comp.hkbu.edu.hk

Zihan Jia
Hong Kong Baptist University
cszhjia@comp.hkbu.edu.hk

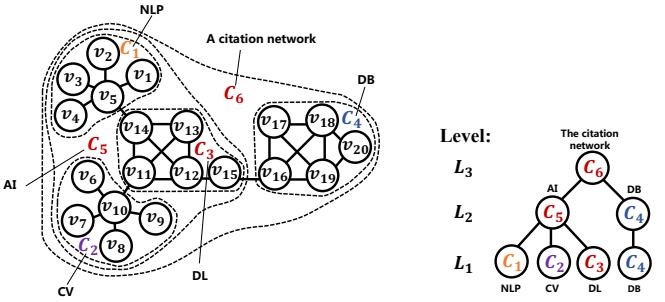
Xin Huang
Hong Kong Baptist University
xinhuang@comp.hkbu.edu.hk

***Abstract*—Communities, formed by a subset of vertices that are densely connected to each other and loosely connected to outside community members, widely exist to represent functional modules in real-world complex systems. Most existing community detection and search methods aim at finding communities at one single level, neglecting the natural properties of overlapping and hierarchy in communities. Therefore, the discovery of hierarchical graph clustering (HGC) to find communities at different levels, which is particularly useful in many applications. However, existing HGC studies suffer from two significant limitations: 1) inefficiency over large-scale networks, and 2) generating too many levels of community hierarchy without distinguishing the hierarchy differences.**

To address the above limitations, we revisit the problem of hierarchical graph clustering and formulate the problem based on our proposed three important properties. To tackle it, we propose theoretical-guaranteed fast solutions, in terms of algorithm complexity and hierarchy levels. We first formulate our HGC problem to admit three key properties of hierarchical communities. Based on the natural hierarchical structure of k-core, we develop a simple and importantly useful technique of *core propagation*. The key idea of core propagation is to take each k-core as one seed of hierarchical communities and find disjoint communities within k-core using a linear-time algorithm of label propagation. We propose two core propagation approaches of top-down and bottom-up algorithms, in terms of different search directions of k-cores by increment and decrement on k, respectively. The top-down method can find a given level of hierarchical communities in $O(tm)$ time, where t is an input of hierarchy levels and m is the graph size. To dismiss the hardness of users' input hierarchy parameter t, the bottom-up algorithm is equipped with a well-designed strategy of auto-adjusting hierarchical levels based on the graph structure itself. We also develop the coreness weight-based label propagation to ensure the accurate label voting of compressed communities at low levels. The bottom-up method runs fast in $O(m \log \delta(G))$, where $\log \delta(G)$ is a small value of the maximum coreness in graph G. Extensive experiments conducted on real-world graphs with ground-truth HGCs validate the effectiveness and efficiency of our proposed core propagation methods against state-of-the-art methods. Two case studies on the world-wide flight network and the Hong Kong road network demonstrate the particular usage of our HGC methods.

I. Introduction

Graph is one of the fundamental data types used for representing relationships between different entities. It's of vital importance to perform analysis and understand the topology of graphs. Community structures are one of the most commonly-seen topological structures in real-world graph datasets such as social networks, citation networks, etc. It's a natural property



(a) A citation network G in AI field and the hierarchical communites on G

(b) HGC dendrogram

Fig. 1. A citation network G and the hierarchical communites on G. Fig. (a) highlights the communities $C_1, C_2, C_3, C_4, C_5, C_6$ using dash circles, where the community $C_1 = \{v_1, v_2, v_3, v_4, v_5\}$. Fig. (b) shows the dendrogram of three-level community hierarchy, where $L_1 = \{C_1, C_2, C_3, C_4\}$, $L_2 = \{C_4, C_5\}$, and $L_3 = \{C_6\}$.

that vertices in the same community have denser connections while vertices in different communities have sparser connections. Community detection methods [20], [34]–[36] provide an effective way to identify all communities in a graph, which enables us to analyze the topological structures in the graph in a more intuitive way. It has very broad applications on market segmentation, friend recommendation, product promotion, etc.

Community detection methods aim to divide a graph into several disjoint sets of vertices based on measuring some properties of the resulting communities. Typical methods include dense subgraph mining based on cohesive subgraph models, modularity maximization, statistical inference, etc. These methods can discover communities only in the same level.

However, communities often exhibit hierarchical structures, e.g. small communities are included in a large community in different level. Discovering such hierarchical structures is beneficial to a better understanding of the network composition and has many real-world applications. For instance, consider a citation network G as shown in Fig. 1(a). Each vertex represents a publication in specific fields, while each edge represents a citation connection. The publications form communities of different scales if we inspect them from different levels. In Fig. 1 (a), the communities in G are highlighted using dashed circles, including subgraphs C_1, C_2, C_3, C_4, C_5, and C_6, where each community stands for a particular research field, e.g. C_1 contains the publications from natural language

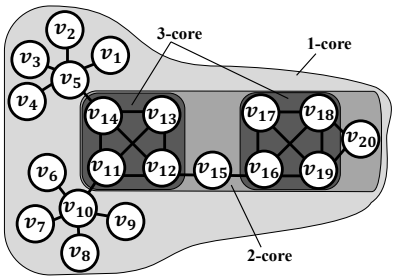


Fig. 2. An example of graph G. The result of core decomposition has three k-cores, i.e., 1-core, 2-core, and 3-core. The maximum coreness $\delta(G) = 3$.

processing (NLP), C_2 contains the publications from computer vision (CV), etc. The research communities form a natural hierarchical structure in terms of research fields at different scale. For example, the entire citation network G contains two main research communities of AI and database as C_5 and C_4, i.e., $C_6 = C_4 \cup C_5$. Furthermore, the AI community C_5 contains three sub-field communities: the NLP community C_1, the CV community C_2, and the deep learning (DL) community C_3, i.e. $C_5 = C_1 \cup C_2 \cup C_3$. Based on these fact, a community hierarchy of graph G can be built as shown in Fig. 1 (b). The self-contained community hierarchy offers a clear view of not only the connections between small research fields, but also the composition of research fields of different scales. Consider a scenario that a researcher is conducting a survey on a particular research field, a clear community hierarchy of research fields can provide efficient ways and important information for him/her to explore different scales of research fields. The core propagation based hierarchical graph clustering framework proposed in this paper focuses on producing effective and elegant community hierarchies, which has wide application to similar scenarios, such as web graphs, social networks, road networks, supply chain networks, and so on. Compared to hierarchical clustering, a significant limitation of traditional community detection methods is ignoring the natural composition of communities, and failing to offer users the opportunity of inspecting communities in different scales. Thus, an efficient and accurate hierarchical graph clustering solution is needed.

To address this issue, hierarchical community detection methods are proposed by some previous studies. According to the exploring direction, the hierarchical community detection methods can be divided into two approaches: 1) Agglomerative (bottom-up) approach; and 2) Divisive (top-down) approach. Agglomerative approaches start with regarding each vertices as an individual communities. It continuously merges the most 'similar' communities to build up the community hierarchies. On the contrary, divisive approaches start by regarding the entire graph as the root hierarchy of communities. And it continuously divides the communities in higher levels into sub-communities to build up hierarchies in a top-down manner. Both of the approaches will produce a dendrogram as a computational result. However, traditional methods of both agglomerative or divisive approaches are all based on some optimization objectives such as modularity-based optimization [10], [34], [42], similarity-based optimization [1], [18], [40], [43], matrix factorization [9], [13], flow-based optimization [38], etc. A recent work Paris [4] builds up the community hierarchies by edge-sampling, which produces massive number of hierarchies and encounters high computational complexity. To obtain community hierarchies with more condensed structure, hLP [39] directly applies the simple label propagation technique to agglomeratively group communities, which produces hierarchical clustering results based on random order of propagation. In summary, the above existing approaches suffer from two limitations:

1)High computational complexity: Since most of the approaches apply exhaustive policies for searching two communities to merge or divide, they often result in high time complexity and weak scalability, e.g. modularity-based methods, similarity-based method, etc. ***2)Too many levels of hierarchies:*** As we have mentioned above, most of the agglomerative approaches merge only two communities in one hierarchy, which may output a dendrogram with complex structure and a large depth. Such kind of structures have no benefits for our analysis of hierarchical communities, since in practical cases, a large community may contain several small sub-communities (often more than two).

Motivated by the example above and to address the above limitations, we propose two core propagation approaches for the hierarchical graph clustering problem. The core idea is to leverage the advantages of the k-core decomposition and the label propagation algorithm to obtain clustering results with elegant hierarchical structures and near linear efficiency. Specifically, k-core decomposition [7] can decompose the input graph into different hierarchies with self-containment nature. Fig. 2 also shows the core decomposition result of G. k-core subgraph with respective different values of k is depicted in different dark regions. However, only adapting the k-core decomposition can produce inaccurate hierarchical clustering results, because it lacks of the ability to break loosely connected ties. For example, the 2-core subgraph in Fig. 2, it fails to correctly separate two densely connected structures formed by $\{v_{11}, v_{12}, v_{13}, v_{14}\}$ and $\{v_{16}, v_{17}, v_{18}, v_{19}, v_{20}\}$, which needs special technique such as label propagation to achieve the goal. Moreover, it's hard to assign community labels to vertices excluded from the current k-core subgraph. For example, int the 2-core subgraph in Fig. 2, community formed by v_1, v_2, v_3, v_4, v_5 can not be accurately identified, which also needs help from some clustering algorithm such as LPA. In our top-down core propagation approach, for core subgraph of each k, we first apply label propagation to separate the k-core subgraph into distinct candidate communities. Secondly, for vertices not belong to the current k-core subgraph, we propose a breath-first search based label propagation to attach them to the closest community. Thirdly, we propose an information gain based strategy to select t key layers of core graph as candidate community hierarchies, where t is an user input integer. Since users may have limited knowledge to new datasets, its hard for users to choose a proper value of t in this case. Therefore, we further propose a parameter-free

bottom-up core propagation approach to automatically obtain all hierarchical communities. Equipped with a community refinement strategy, the number of hierarchies can be bounded by $\log \delta(G)$, where $\delta(G)$ is the degeneracy of the input graph. To summarize, our contributions can be listed as follows.

- We define an important property of community hierarchy and formulate the problem of hierarchical graph clustering. (Section III)
- We leverage the concept of k-core and the label propagation technique to design a top-down core propagation approach with an user input parameter t, which can obtain high quality hierarchical clustering results with t hierarchies with time complexity near linear to the graph size. (Section V)
- We further propose a parameter-free bottom-up core propagation approach equipped with a weighted label propagation to produce hierarchical clustering results with $\log \delta(G)$ levels and $O(m \log \delta(G))$ time. (Section VI)
- We validate the efficiency of our proposed method and propose two new evaluation metrics to compare the effectiveness of all methods. Moreover, we conduct two case studies on real-world ground-truth datasets to demonstrate the applications of our methods. (Section VII)

The rest of this paper is organized as follows. We discuss the related works in Section II. We give an overview of the entire core propagation framework in Section IV. Finally, we conclude the contributions of this paper in Section VIII.

II. Related Work

This work is mostly related to k-core mining, communuity detection, and hierarchical community detection.

K-Core mining. There exist 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 [6]. 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 [7], streaming algorithms [41], distributed algorithms [32], and I/O efficient algorithms [44]. The study of core decomposition is also extended to different types of graphs such as dynamic graphs [2], [22], uncertain graphs [5], directed graphs [16], [26], temporal graphs [45], and multi-layer networks [19]. Recently, core maintenance in dynamic graphs has attracted significant interest in the literature [2], [28], [47]. In addition, several k-core based community models have been proposed for community search [14], [15], [17], [29]. The k-core decomposition is also applied to graph clustering problem. Inspired by the idea of spectral clustering, Mei et al. [30] propose the k-core motif to produce primary clusters, and assign the remaining vertices to the closest clusters. Using the similar idea, Giatsidis et al. firstly treat the k-core subgraph with the maximum coreness to be primary clusters, and assign the vertices with lower coreness to the known cluster according to a probability function. These works ignore a fact that vertices in the same k-core subgraph may contain loosely connected edges, which can be further divided into different partitions. Furthermore, the above two works only focus on producing flat clusters in the same level.In this paper, we leverage the LPA technique to both separate vertices in the same core subgraph into distinct clusters and propagate label information to the remaining vertices. Moreover, we focus on producing hierarchical community result instead of graph clustering in a single level.

Community detection. The problem of community detection aims to identify all dense communities in a graph. Traditional community detection methods often focus on maximizing some quality measures of the identified communities such as modularity [34], betweenness centrality [20], etc. However, these methods often exhibit high computational complexities, which is not scalable for large graphs. For example, the modularity maximization based approach is an exhausted method that compares the modularity over all possible divisions, which is intractable. Other methods include spectral methods [35], random walk based methods [36], etc. Besides finding disjoint communities, some studies [23], [27], [33] focus on finding overlapping communities. Recently, Guan et al. propose a non-negative matrix factorization based approaches to take into consideration the higher order network topological structures for community detection. The above studies only focus on one layer of the input graph. In this paper, we aim to discover a hierarchy of communities.

Hierarchical community detection. The goal of the hierarchical community detection problem is to find a community hierarchy for an input graph. Existing works are usually based on different optimization objectives. Modularity-based approaches like FN [34], Clauset [10] and Toujani [42] keep merging clusters with greatest modularity increase. But they suffer from a complexity of $O(mdlog(n))$. Similarity-based approaches like SingleLink [1], Paris [4], hLP [39] and MST [43] merges the clusters with highest common neighbor based similarity in each hierarchy. But most of them encounter high computation complexity such as $O(n^2)$. Matrix factorization based approaches like Spec [9] and NMF [13] perform matrix factorization on the Laplacian matrix of the original graph. However, they are running in the highest time complexity of $O(n^3)$. In recent years, several methods based on game theory [48], random walk [46], min-cut [38] and divisive similarity [18], [40] are proposed. But all the above methods suffer from high computation overhead. Moreover, most of them produce massive number of hierarchies, which makes the clustering hard to understand by users. Recently, a few studies [11], [12], [31] focus on producing flat hierarchy by dendrogram flattening with different problem setting. Monath et al. [31] study the hierarchical clustering problem on vector data. Another two solutions in [11], [12] study the hierarchical graph clustering problem on graphs. But they require weighted graph as input and mainly focus on proposing distributed parallel algorithms. Different from the above works, in this paper, we aim at proposing sequential algorithms for finding high quality community hierarchies on unweighted simple graph with at most $\delta(G)$ levels with nearly

linear time complexity with respect to the graph size.

III. Preliminaries

In this section, we introduce useful preliminaries and our problem formulation. We consider an undirected and simple graph $G(V, E)$ where V is the set of n vertices and E is the set of m edges, i.e. $|V| = n$ and $|E| = m$. For a vertex $v \in V$, the set of v's neighbors is denoted as $N(v) = \{u \in V : (v, u) \in E\}$. The degree of v is denoted as $deg(v) = |N(v)|$, and the maximum degree in graph G is $deg_{max} = \max_{v \in V} deg(v)$. For a subset S of vertices, the induced subgraph H of G by vertices S is represented as $H = (V(H), E(H))$, where $V(H) = S$ and $E(H) = \{(v, u) \in E : v, u \in S\}$. The degree of v in subgraph H is $deg_H(v)$. Based on the minimum degree in a subgraph H, we introduce two useful definitions of k-core and coreness as follows.

Definition 1: **(k-core)** [8] For an integer $k \in \mathbb{Z}^{\geq 0}$, a k-core H is a subgraph of G such that each vertex has at least k neighbors within H, i.e. $deg_H(v) \geq k$.

The core decomposition process follows a peeling strategy that keeps removing the vertex with the smallest degree to ensure the remaining subgraph to be a k-core. The process start from 1 to the largest value of k, which computes the k-core subgraph for all possible values of k. Consider a graph G shown in Fig. 2. It depicts the core decomposition result of k-cores for all possible values k in graph G. The entire graph is a 1-core, which is highlighted by the largest gray area. The 2-core is highlighted by a darker gray area, including vertices $v_{11}, v_{12}, v_{13}, ..., v_{20}$. The 3-core is highlighted by two darkest gray area, including vertices $v_{11}, v_{12}, v_{13}, v_{14}, v_{16}, v_{17}, v_{18}, v_{19}$. As we can see, the organization of all k-cores is represented in a hierarchical structure. That is, 3-core is contained in the 2-core, and moreover in turns to be contained in a large region of 1-core. To be more specific, the coreness of a subgraph $H \subseteq G$ and a vertex $v \in V$ can be defined as follows.

Definition 2: **(Coreness)** The coreness of a given subgraph $H \subseteq G$ is defined as the minimum degree of vertices in H, i.e., $\phi(H) = \min_{v \in H}\{deg_H(v)\}$. The coreness of a given vertex $v \in V$ is defined as $\phi(v) = \max_{H \in G, v \in H} \phi(H)$.

We use $\delta(G)$ to denote the degeneracy of G, which is the maximum coreness among all vertices. And we use H_k to denote the largest k-core or the core subgraph with coreness k in G. In graph G in Fig. 2, the coreness of G is 1, i.e.$\phi(G) = 1$. The degeneracy of G is $\delta(G) = 3$.

In a graph G, the community C is usually a dense subgraph such that vertices are densely connected to others within C but have few connections to others outside of C. For simplicity, we use C represented by the set of vertices in V, i.e., $C_i \subseteq V$. Based on the communities, we give a definition of graph partition.

Definition 3: **(Graph Partition)** A partition L of a graph G is a set of $r \in \mathbb{Z}^+$ *disjoint communities*, and the union of r communities is the whole vertex set V, e.g., $L = \{C_1, C_2, \ldots, C_r\}$ s.t. $\bigcup_{i=1}^{r} C_i = V$ and $\forall i, j \in \{1, 2, ..., r\} \wedge i \neq j$, $C_i \cap C_j = \emptyset$.

For example, in graph G in Fig. 1 (a), C_1, C_2, C_3, C_4 can be consider as disjoint communities. A partition L_1 can be formed by $C_1, C_2, ..., C_4$, i.e.,$L_0 = \{C_1, C_2, C_3, C_4\}$. Another partition L_1 can be found by $L_2 = \{C_4, C_5\}$. Communities in a graph often exhibit hierarchical structures. Consider Fig. 1 (b) as an example, a large community C_5 consists of three small communities C_1, C_2 and C_3 in the higher level. Label propagation algorithm (LPA) is a popular method for graph partitioning. In each iteration, each vertex uses the most frequent label of its neighbor as its own label. The LPA process terminates until no further changes of vertex label. Next, we define the hierarchy of communities as follows.

Definition 4: **(Community Hierarchy)** A community hierarchy $\mathcal{L}$ is defined as a set of graph partitions from k levels, i.e., $\mathcal{L} = \{L_1, L_2, \ldots, L_k\}$, where L_i is the partition on i-th level.

Based on the definition of community hierarchy, we use C_p^q to denote the p-th community in L_q partition. To give a non-trivial structural guarantee to the community hierarchy, we propose an important hierarchical property as follows.

Property 1: **(Hierarchical Inheritance)** A community hierarchy is hierarchically inherited *if and only if* the structure of hierarchical communities fulfills the following requirements:
1. Hierarchical Structure. For $1 \leq i \leq j \leq k$, a community at the i-th level L_i should be contained in a large community at the j-th level L_j. Specifically, for each community $C_x^i \in L_i$, there exists a unique community $C_y^j \in L_j$ such that $C_x^i \subseteq C_y^j$ for all $i < j$.
2. Inheritance. For all $1 < i < k$, each community C_x^i in level i is an union set of at least one community in level $i - 1$.
3. Monotonicity. The number of communities should be monotonically increasing from root level to bottom level, i.e. $|L_i| < |L_{i-1}|$ for all $1 < i < k$.

The first two requirements ensure the large communities in higher level should consist of several small communities in the lower level, which is the natural way that multi-scale communities form. Moreover, the third requirement ensures that partitions in different hierarchies can not be identical, which filters meaningless hierarchies.

In this paper, we study a new problem of hierarchical graph clustering (HGC-problem). Different from previous studies [4], [39], the aim of our HGC problem is to identify *a small number of hierarchies* to naturally represent all communities in the organization of *same-level disjoint* and *cross-level self-contained*. Community results at different leveled hierarchies significantly distinguish from others. Therefore, we formulate the problem in detail as follows.

Problem 1: **(Hierarchical Graph Clustering)** Given a graph $G = (V, E)$, the goal of hierarchical community detection is to find a community hierarchy $\mathcal{L}$ satisfying Property 1 of hierarchical inheritance property.

Example 3.1: Given graph G in Fig. 2 as input, a feasible hierarchical graph clustering result $\mathcal{L}$ is shown in Fig. 1 (a) and (b). $\mathcal{L}$ contains three hierarchies L_1, L_2 and L_3, where $L_1 = \{C_1, C_2, C_3, C_4\}$, $L_2 = \{C_4, C_5\}$ and $L_3 = \{C_6\}$. We can observe that $\mathcal{L}$ strictly fulfills Property 1. For the first

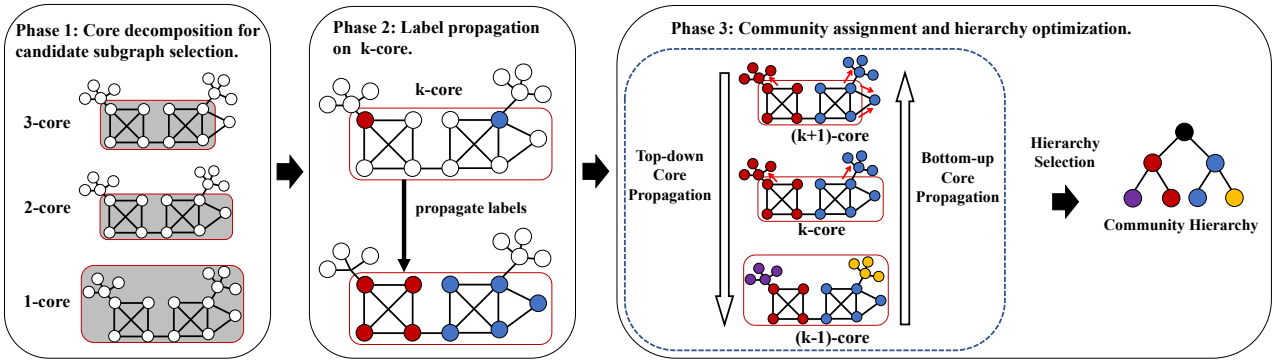


Fig. 3. The framework of our core propagation based hierarchical graph clustering with three key phases in terms of top-down and bottom-up manners.

requirement, all communities in $\mathcal{L}$ have unique parent communities in its higher level, e.g. $C_1 \subseteq C_5 \subseteq C_6$. For the second requirement, each community in higher level is a unique union of communities in its next level, e.g. $C_6 = C_4 \cup C_5$ and $C_5 = C_1 \cup C_2 \cup C_3 \cup C_4$. For the final requirement, the number of communities is monotonically increasing from root level to bottom level, e.g. $|L_3| < |L_2| < |L_1|$.

IV. An Overview of Core Propagation Framework

To solve the hierarchical graph clustering problem, we first propose a top-down core propagation approach that utilizes an user input parameter t to produce hierarchical communities with t hierarchies. Furthermore, we then provide a parameter-free bottom-up propagation approach to automatically obtain hierarchical communities with $\log \delta(G)$ hierarchies. They share the similar key idea of core propagation. A general framework is shown in Fig. 3 and can be summarized into three phases as follows:

- **Phase 1:** Core decomposition for the seed community selection. In this phase, the core decomposition algorithm is applied to incrementally obtain k-core subgraph for all k values. The k-core subgraphs are treated as candidate communities in each hierarchy.
- **Phase 2:** Label propagation on candidate k-cores. In this phase, label propagation process is applied from different directions to split or group communities. For the top-down core propagation approach, it retrieves the k-core subgraph for a particular k as candidate subgraph in each level. Label propagation process is then conducted to separate the candidate subgraph into distinct initial communities. For the bottom-up core propagation approach, a weighted label propagation process is conducted on all vertices considering the coreness weights of each vertex during the label counting to group closely connected communities.
- **Phase 3:** Community assignment for unlabeled vertices and hierarchy optimization. In this phase, community labels will be assigned to all unlabeled vertices. For the top-down approach, vertices without labels in the same layer will be attached to the closest communities using a breadth-first search scheme. For the bottom-up approach, vertices with the same label will be grouped into super nodes, and super edges are added to form a super graph in the next level. To optimize the number of hierarchy, an information gain based technique is proposed to select proper hierarchies for label propagation, while a in-layer community refinement strategy is designed to ensure the number of the hierarchies for the bottom-up approach.

V. Top-down Core Propagation

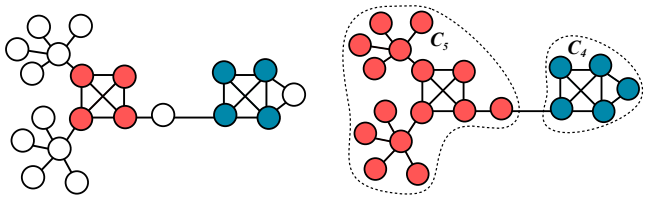
In this section, we develop a top-down core propagation approach to solve the hierarchical graph clustering problem and analyze its computational complexity.

A. Exhausted Top-down Core Propagation

The top-down core propagation approach first leverages the k-core subgraph for all possible values of k as candidate community in each layer, where each layer corresponds to a particular value of k. In each layer, label propagation process is conducted on the candidate k-core subgraph to split them into different communities. For vertices with coreness smaller than the current k, their community labels are assigned by the nearest community in a breadth-first search manner. This strategy starts from the entire graph and iteratively treats the k-core subgraphs with descendant coreness as candidate subgraph to create new clusters in the lower levels. Moreover, to condense the community hierarchy, we introduce a parameter t and develop a top-t candidate hierarchy selection scheme based on information gain. In the following, we introduce each technique in detail.

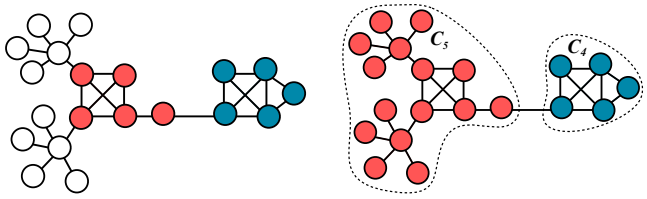
Candidate core subgraph identification. The core decomposition algorithm is applied to identify the k-core subgraph for all possible values of k. Each core subgraph with a particular coreness of k is regarded as the candidate community structure in each layer. This is because each k-core itself is a cohesive subgraph, which exhibits initial community feature in each layer. Moreover, the k-core subgraph has a self-containment nature for different values of k, which exhibits hierarchical community feature.

Community identification and label assignment. The top-down core propagation approach starts at the subgraphs with the largest coreness $\delta(G)$. Label propagation is performed among the k-core subgraph to split them into different communities first, and then attach the vertices with smaller coreness to



(a) Label propagation on 3-core. (b) BFS core propagation.

Fig. 4. The core propagation process on 3-core in the top down approach.



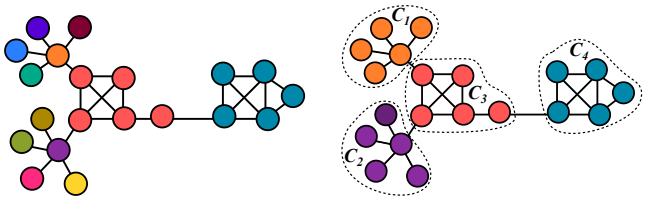
(a) Label propagation on 2-core. (b) BFS core propagation.

Fig. 5. The core propagation process on 2-core in the top down approach.

the closest classified communities using a breadth-first search manner. In the next hierarchy, the $(\delta(G)-1)$-core subgraph is considered. Vertices appear in both $(\delta(G)-1)$ and $\delta(G)$ subgraphs will inherit labels from the last hierarchy. Then the same label propagation and BFS process are conducted again to obtain a new clustering result. This procedure is repeated until the k-core subgraph with smallest coreness is processed. Finally, the hierarchical graph clustering result is obtained after all iterations are completed.

Algorithm. The details of the top-down core propagation approach is shown in Algorithm 1. It first applies core decomposition on G to obtain core subgraphs H_k with all possible coreness values k (lines 2-3). In the next step, it utilizes the LPA to detect hierarchical communities for each hierarchy (lines 8-18). In each iteration, it retrieves an candidate core subgraph $\mathcal{H}_i$ (line 8). For the vertices that has appeared and classified in $\mathcal{H}_{i-1}$ in the last iteration, it inherits the community labels in L_{i-1} (lines 9-10). For the other vertices in $\mathcal{H}_i$, it assigns new labels to them (lines 11-12). Next, it applies label propagation process on $\mathcal{H}_i$ to split the $\mathcal{H}_i$ into different communities (line 13). For the remaining vertices in the input graph G excluding $\mathcal{H}_i$, it just assigns them with the labels of the nearest communities (lines 14-15). Next, it retrieves the communities in the i-th hierarchy according to the community labels of vertices (line 16). To ensure the monotonicity property, it first checks whether the clustering results in L_i is the same with its parent layer L_{i-1}. If yes, the result of L_i will be stored in $\mathcal{L}$ (line 17). To ensure the hierarchical inheritance property in Property 1, it needs to remove the edges between the identified communities (line 18). Finally, it returns the community hierarchy $\mathcal{L}$ as output.

Example 5.1: Taking graph G in Fig. 2 as an example, it starts with the k-core subgraph with maximum coreness 3. As shown in Fig. 4 (a), the vertices in 3-core are initialized with different labels first. Second, the label propagation process is conducted to separate the 3-core into two communities which are highlighted by red and blue colors. Thirdly, in



(a) Label initialization on 1-core. (b) Label propagation on 1-core.

Fig. 6. The core propagation process on 1-core in the top down approach.

Fig. 4 (b), a breadth-first search process will start from the classified vertices (vertices in red and blue colors) to search the unlabeled vertices (vertices in white color) to assign their community labels to them. This process will be repeated until all the vertices are labeled.. In this result, C_4 and C_5 is identifies as two distinct communities, which is the same clustering result in the second level of community hierarchies in Fig. 1 (b). Next, in Fig. 5, it continues to process the 2-core subgraph. The vertices belong to both 3-core and 2-core are assigned the same labels in the last hierarchy. For the other vertices in 2-core, they are assigned with new labels. Fig. 5 (a) shows the label propagation result among the 2-core. In Fig. 5 (b), BFS is conducted to obtain the graph clustering result of the current hierarchy. Core propagation in this level produces same result as the last level, e.g. Fig. 4 (b). This result can be condensed by the hierarchy condensing optimization strategy introduced in next subsection. Finally, it continues to process the 1-core subgraph. In Fig. 6(a), all vertices belongs to 2-core are assigned the historical labels from Fig. 5 (b). Other vertices with coreness 1 are assigned new labels. After label propagation, three new communities C_1, C_2 and C_3 are obtained and highlighted in orange, purple and red colors in Fig. 6(b), which is the same result as the first level of community hierarchies in Fig. 1 (b).

B. Hierarchy Condensing Optimization

In practical cases, $\delta(G)$ is still a large number for exploring the community hierarchy. Moreover, the largest k-core and the largest $(k-1)$-core may remain the same, i.e., $H_k = H_{k-1}$, or just incur small difference because of the self containment property of k-core, i.e., $H_k \subseteq H_{k-1}$. For example, the 2-core subgraph in Fig. 2 has only two extra vertices compared to the 3-core subgraph. Performing splitting process among them will not be beneficial from the perspective of information theory. The clustering results in Fig. 4 (b) and Fig. 5 (b) are the same, which confirms this analysis. To reduce the height of community hierarchy, we introduce a user input parameter $1 \leq t \leq \delta(G)$ to condense the hierarchy number to t.

Based on the input parameter t, we are interested to find t community hierarchies with greatest difference. As mentioned before, the difference between two largest cores with different k values can provide a simple yet useful intuition to the difference between two hierarchies. We quantify such difference between two adjacent largest core with coreness k and $k-1$ as information gain. The definition of the information gain for a particular core subgraph with respect to coreness k is given as follows.

Algorithm 1 Top-down Core Propagation

Input: A graph $G = (V, E)$, and a hierarchy parameter t.
Output: Community Hierarchy $\mathcal{L}$.

1: $\mathcal{L} \leftarrow \emptyset$, the candidate core subgraph $\mathcal{H} \leftarrow \emptyset$;
2: $H \leftarrow$ Compute k-core subgraphs by core decomposition;
3: $\delta(G) \leftarrow \max_{v \in V} \phi(v)$;
4: **for** k from $\delta(G)$ to 1 **do**
5: $\quad IG(k) = \frac{|H_k| - |H_{k+1}|}{H_k}$;
6: $\mathcal{H} \leftarrow$ top-t core subgraphs with the highest IG;
7: Sort $\mathcal{H}$ w.r.t. coreness;
8: **for** each $\mathcal{H}_i = (V_{\mathcal{H}_i}, E_{\mathcal{H}_i}) \in \mathcal{H}$ **do**
9: $\quad$ **for** each vertex $v \in V_{\mathcal{H}_{i-1}}$ **do**
10: $\quad\quad label_i(v) \leftarrow label_{i-1}(v)$;
11: $\quad$ **for** each vertex $u \in V_{\mathcal{H}_i} / V_{\mathcal{H}_{i-1}}$ **do**
12: $\quad\quad label_i(u) \leftarrow$ New label;
13: $\quad \mathcal{H}_i \leftarrow$ Label propagation on $\mathcal{H}_i$;
14: $\quad R \leftarrow G - \mathcal{H}_i$;
15: $\quad$ **for** each $w \in R$ **do** $label_i(w) \leftarrow$ Label assignment by BFS;
16: $\quad L_i = \{C_1^i, ..., C_r^i\} \leftarrow$ Obtain the i-th level of communities;
17: $\quad$ **if** $|L_i| > |L_{i-1}|$ **then** $\mathcal{L} = \mathcal{L} \cup L_i$;
18: $\quad$ Remove the edges between different communities;
19: **return** $\mathcal{L}$;

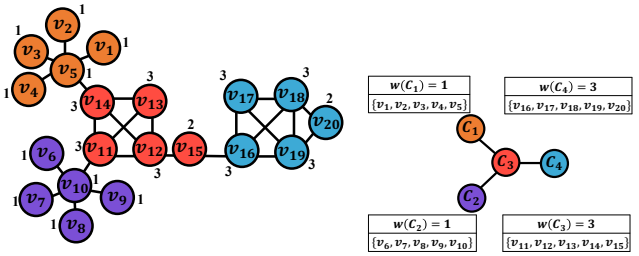
Definition 5: (Information Gain) The information gain of the largest core subgraph with coreness k is defined as

$$IG(k) = \frac{|H_k| - |H_{k+1}|}{H_k}$$

Hence, the idea of the hierarchy condensing optimization is to select t largest core with greatest information gain as candidate hierarchies. In this manner, the candidate hierarchies with the same or similar core sugraph structures will be filtered out because their information gain will approaches 0. Afterwards, The top-down core propagation is performed on the candidate hierarchies to obtain the final hierarchical clustering result.

The details of hierarchy condensing optimization are shown in lines 4-7 in Algorithm 1. After core decomposition, it continues to compute the information gain for all core subgraphs H_k according to Definition 5 (lines 4-5). Next, it stores t number of core subgraphs with highest information gain into $\mathcal{H}$ as candidates for label propagation, and sorts them in descending order according to their coreness (lines 6-7).

Example 5.2: Consider graph G in Fig. 2, the number of vertices in its 3-core subgraph is 8, hence $|H_3| = 8$. Similarly, we have $|H_2| = 10$, $|H_1| = 20$. By Definition 5, since $|H_4| = 0$, $IG(3) = \frac{|H_3| - |H_4|}{|H_3|} = \frac{8-0}{8} = 1$. In the same way, we have $IG(2) = \frac{|H_2| - |H3|}{|H_2|} = 0.2$, and $IG(1) = \frac{|H_1| - |H_2|}{|H_1|} = 0.5$. Assuming that the hierarchy parameter t is 2, the top-2 core subgraphs with highest information gain is H_3 and H_1. Therefore, H_3 and H_1 are selected as candidate core subgraph. After performing core propagation on H_3 and H_1, the clustering result in Fig. 4 (b) and Fig. 6 (b) will be the final community hierarchies, which is the same as the community hierarchies shown in Fig. 2. The result in Fig. 5 (b) is ignored, which produces a more condensed hierarchical structure.



(a) Weighted label propagation. (b) Super graph construction.

Fig. 7. The core propagation process on the bottom up approach.

Complexity analysis. We consider a grpah $G(V, E)$ for $|V| = n$ and $|E| = m$. Without loss of generality, we assume that $m \leq n - 1$ for simplifying complexity analysis [24]. We analyze the time and space complexity of Algorithm 1 in the following theorem.

Theorem 1: The top-down approach in Algorithm 1 takes $O(tm)$ time and $O(tn + m)$ space.

Proof: We first analyze the time complexity of Algorithm 1. The core decomposition to obtain all k-core subgraphs takes $O(m)$ time (line 2). Since the number of k-core subgraphs is $\delta(G)$, computing the top-t core subgraphs with the largest information gains can be done in $O(\delta(G) \log t)$ time, using a t-sized minimum heap (lines 4-6). Sorting candidate core subgraphs in the top-t results cost $O(t)$ time using a bin sort mechanism. In each level, the label propagation process run on a core subgraph takes $O(m)$ time [37], and the breadth-first search based label assignment for unlabeled vertices takes also $O(m)$ time. Since the number of hierarchy is t, the same process should be executed for t times, which costs $O(tm)$ time in total to process all hierarchies. Therefore, the time complexity of Algorithm 1 is $O(m + \delta(G) \log t + tm) \subseteq O(tm)$, as the input parameter is usually $t \leq n$ and $\delta(G) \leq n - 1 \leq m$ in the setting.

Next, we continue to analyze the space complexity. First, the core decomposition and information gain based core subgraph selection can be done in $O(m)$ space. Secondly, the entire label propagation process in all levels takes $O(m)$ space. In each level, the community result is stored in $O(n)$ space. Since it has t levels of hierarchies, it takes $O(tn)$ space to store the entire community hierarchy. Overall, the space complexity of Algorithm 1 is $O(tn + m)$.

VI. Bottom-up Core Propagation

The advantage of the top-down core propagation approach is that it includes an input parameter of t to control the number of the resulting community hierarchies. However, it's difficult for users to choose a proper value of t when dealing with unknown data, which is inflexible. To address this, we propose a bottom-up core propagation approach to automatically obtain effective community hierarchies without any input parameters in this section.

The bottom-up core propagation is an agglomerative method starting from the bottom layer, which treats each vertex as an

Algorithm 2 Bottom-up Core Propagation

Input: $G = (V, E)$
Output: The community hierarchy $\mathcal{L}$

1: $i \leftarrow 1$;
2: $G_i = (V_i, E_i) \leftarrow G$;
3: **while** $|V_i| > 2$ **do**
4: **for** each vertex $v \in V_i$ **do**
5: $weight(v) \leftarrow \max_{u \in v, u \in V} \phi(u)$;
6: $\mathcal{LB} \leftarrow$ Apply Algorithm 3 on G_i;
7: $L_i \leftarrow$ Obtain communities w.r.t $\mathcal{LB}$;
8: $G_{i+1} = (V_{i+1}, E_{i+1}) \leftarrow$ Super graph construction w.r.t L_i;
9: **while** $(|V_i| - |V_{i+1}|) \leq \frac{|V_i|-1}{\log \delta(G)-i}$ **do**
10: $G_{i+1} = (V_{i+1}, E_{i+1}) \leftarrow$ Community refinement;
11: $i \leftarrow i + 1$;
12: **return** $\mathcal{L} = \{L_1, L_2, ..., L_i\}$;

Algorithm 3 Weighted Label Propagation

Input: $G = (V, E)$, an iteration number L
Output: Labels of the vertices $\mathcal{LB}$

1: $\mathcal{LB} \leftarrow \emptyset$;
2: **for** each $v \in V$ **do**
3: $Label(v) \leftarrow v$;
4: $\mathcal{LB} \leftarrow Label(v)$;
5: **for** i from 1 to L **do**
6: Shuffle all the elements in V;
7: **for** each $v \in V$ **do**
8: $Label(v) \leftarrow$ Apply Procedure GetMaxLabel on v;
9: **if** $\mathcal{LB}$ remains unchanged **then** break;
10: **return** $\mathcal{LB}$;

Procedure GetMaxLabel(v)

1: **for** each neighbor u of v **do**
2: $Freq(Label(u)) \leftarrow$ Accumulate the weight of u $weight(u)$;
3: $label \leftarrow \arg\max_{Label(u) \in \{Label(u): u \in V\}} Freq(Label(u))$;
4: **return** label;

individual community. Label propagation process is applied for clustering them into different communities in one hierarchy. In each iteration, super nodes will be used to represent vertices in the same communities. And super edges are used to represents the cross-community connection in the previous iteration. The LPA will be repeated on the super graph induced by the new super nodes and super edges until the entire graph are merged into one super node. To reduce the height of the resulting community hierarchy, we propose a weighted version of LPA. And the weights are obtained by coreness information from core decomposition.

Alogorithm. The detail steps of the bottom-up core propagation approach are listed in Algorithm 2. Firstly, it treats the original graph G as a super graph G_1 used in the first iteration (lines 1-2). Secondly, in each iteration, for each super node v in V_i, it assigns the maximum coreness of the vertices inside itself as the weight of v, i.e., $weight(v) = \max_{u \in v, u \in V} \phi(u)$ (lines 4-5). Next, it applies the weighted label propagation algorithm to merge the super nodes into new communities, and obtain community labels for all super nodes in G_i (line 6). The details of weighted label propagation are shown in Algorithm 3. The main process of this algorithm is very similar to the unweighted version of LPA. The only different is the label calculation for each vertex, which is shown in the getMaxLabel procedure. Traditionally, a vertex will adopt the labels from its neighbors with the highest frequency. Here, it counts the frequency of a label from its neighbor using the weights of this neighbor (line 2 in Procedure getMaxLabel). This small change can attach looser structures to denser structures in a faster way, which also to some extent avoids the sensitivity caused by random seed selection. Algorithm 2 continues to retrieve communities at the i-th hierarchy according to the computed community labels (line 7). Then, it constructs super graph G_{i+1} according to the community structures identified in G_i line 8). The entire process will repeat until all the nodes are merged into one super node (line 3). Finally, it returns the community hierarchy $\mathcal{L}$ as output.

Example 6.1: As shown in Fig. 7 (a), using the graph G as an example, the coreness is assigned as weight of each vertex. A weighted label propagation is conducted among the weighted graph G, the clustering result is highlighted using different colors in Fig. 7 (a). According to the result of the first layer, vertices with same labels are grouped into super nodes as shown in Fig. 7 (b), and new weights are assigned to super nodes. As shown in Fig. 7 (b), label propagation is run again on the super graph to obtain the final result.

Community refinement in same level. To ensure the brevity of the hierarchy and the efficiency of the algorithm, we propose to refine the community results in the same level. Since the idea of the bottom up approach follows an automatic strategy to group the communities in the same level to form a new hierarchy, which can still produce deep community hierarchies with unbounded number of levels. To provide a theoretical guarantee on the number of levels, we combine the communities with most inter-edges into a new one until the decrease of community numbers between two levels are bounded. As shown in lines 9-10 in Algorithm 2, we manually combine the communities with most inter-edges to refine the results in the same level. Specifically, given the decrease of community number between the current level and the previous level $|v_i| - |v_{i+1}|$, if this value is smaller than a dynamic threshold denoted as $\frac{|V_i|-1}{\log \delta(G)-i}$, the communities in L_i should be refined. Under the community refinement strategy, the monotinicity property can also be ensured.

Complexity analysis and discussions.

Theorem 2: The levels of community hierarchies obtained by Algorithm 2, i.e. $|\mathcal{L}|$, is bounded by $\lceil \log \delta(G) - 1 \rceil + 1$. Thus, Algorithm 2 takes $O(m \log \delta(G))$ time and $O(m + n \log \delta(G))$ space.

Proof: First, we prove that the main loop part of the algorithm is executed at most $\lceil \log \delta(G) - 1 \rceil$ times. Assuming that it is executed $t = \lceil \log \delta(G) - 1 \rceil$ times i.e. $|V_t| > 2$, we prove that $|V_{t+1}| \leq 2$. Consider that the loop termination condition for the refinement part (lines 9-11) is $(|V_i| - |V_{i+1}|) > \frac{|V_i|-1}{\log \delta(G)-i}$ and $0 < \log \delta(G) - \lceil \log \delta(G) - 1 \rceil \leq 1$, therefore $|V_{t+1}| \leq |V_t| - \frac{|V_t|-1}{\log \delta(G)-t} = |V_t| - \frac{|V_t|-1}{\log \delta(G)-\lceil \log \delta(G)-1 \rceil} \leq |V_t| - (|V_t| - 1) = 1 \leq 2$. So the main loop part will

TABLE I
NETWORK STATISTICS

Name	$\|V\|$	$\|E\|$	deg_{max}	$\delta(G)$
OpenFlight	3,096	18,193	237	31
Facebook	4,039	88,234	1,045	115
Amazon	334,863	925,872	549	6
DBLP	425,957	1,049,866	343	113
Twitter	81,306	1,768,149	3,383	96
road-HongKong	1,067,242	5,582,492	4,029	49
Google	916,428	4,322,051	6,332	44
LiveJournal	4,847,571	34,681,189	14,815	372
Orkut	3,072,441	117,185,083	33,313	252

be aborted at the end of round $t = \lceil \log \delta(G) - 1 \rceil$. Hence The levels of the hierarchical clustering results obtained by Algorithm 2 do not exceed $\lceil \log \delta(G) - 1 \rceil + 1$. And since each execution of Weighted Label Propagation takes $O(m)$ times in $O(n)$ space. Overall, Algorithm 2 takes $O(m \log \delta(G))$ time in $O(m + n \log \delta(G))$ space.

Remarks: Discussion on the selection of top-down and bottom-up approaches. Since the top-down propagation approach includes a parameter t to produce exactly t number of hierarchies, it is more suitable for users with rich empirical knowledge about the input graph data, e.g. the number of the resulting hierarchy is known. However, for some new graph datasets without empirical knowledge, the parameter-free bottom-up core propagation approach is more preferable, since it can obtain hierarchical graph clustering results automatically. In addition, to further accelerate the efficiency, it could explore the parallelization of our proposed algorithms, as both the top-down and bottom-up approaches contain independent computation step. Specifically, for the top-down approach, the label inheritance, the LPA in the same core subgraph, and the BFS based label assignment can be executed in parallel. For the bottom-up approach, the weighted LPA step and the super graph generation can be parallelized.

VII. EXPERIMENTS

In this section, to evaluate the efficiency and effectiveness, we compare our proposed algorithms with the existing competitors on real-world datasets. The code of our algorithms is implemented in C++ with -O3 optimization. All the experiments are run on a machine with dual 6-core 2.66GHz Intel Xeon CPU and 32G memory. The source codes have been publicly available on https://github.com/vldb25code/HGC-Cluster.

Datasets: We run the experiments on nine real-world datasets from web source. There are mainly four types of network data, including flight network (OpenFlight[1]), collaboration network (DBLP [25]), and social networks (Facebook, Twitter, Google, LiveJournal, and Orkut [25]). Moreover, we extract the road network of Hong Kong from the OpenStreetMap[2] platform and reconstruct a road network graph called road-HongKong from it. The network statistics are shown in Table I. We report the number of vertices $|V|$, the number of edges $|E|$, the maximum degree of all vertices deg_{max}, and the maximum coreness as the degeneracy denoted by $\delta(G)$.

Compared methods: In our experiments, we compare the efficiency and effectiveness of our proposed top-down and bottom-up algorithms with four competitors of hierarchical graph clustering methods as follows.

- HCP-t: is our method of top-down core propagation based hierarchical graph clustering proposed in Algorithm 1. For the method HCP-t @t, the input parameter t is used to generate t different levels of graph clusters. We set $t = 3$ by default.
- HCP-b: is our method of bottom-up core propagation hierarchical graph clustering proposed in Algorithm 2.
- hLP [39]: is the hierarchical graph clustering method that applies LPA only to build up community hierarchies.
- Paris [4]: is an agglomerative method to generate $n - 1$ levels of graph clusters, which utilizes the node pair sampling to maximize the modularity-based score during each cluster merging.
- Spec [9]: is a classical spectral method, which leverages the node embedding space to iteratively obtain $n-1$ levels of community hierarchies.
- TeraHac@Seq [12]: is a distributed parallel hierarchical graph clustering algorithm. Here, we test the sequential version of TeraHac@Seq provided by the authors. Since the algorithm requires a weighted graph as input, we only compare the number of cluster hierarchies and the running time.

Evaluation metrics: To evaluate the efficiency, we report the running time of all hierarchical graph clustering methods above. To evaluate the effectiveness, we first compare the number of levels of the resulting community hierarchy. We further propose two evaluation metrics based on the concept of least common ancestor.

- **LCA cost:** To evaluate the clustering effectiveness on the datasets without ground-truth. We propose a new metric based on Least Common Ancestor (LCA).

$$LCAcost = \sum_{e=(u,v)\in E} \frac{LCA(u,v)}{|E| * |\mathcal{L}|}$$

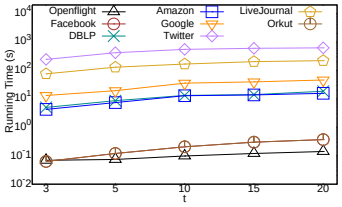
  , where $LCA(u, v)$ is the minimum number of hierarchy vertices u and v can form a community. The idea is based on the assumption that if two vertices u, v are connected by an edge $e = (u, v)$, they will form a community mostly near the leaf level. The smaller the LCAcost is, the better the clustering effectiveness is. Noted that the design of the LCA cost only focuses on the hierarchy and the original graph structure. It does not contain any bias on the clustering techniques.
- **HSS score:** To evaluate the quality of the hierarchical graph clustering on ground-truth datasets, some traditional evaluation metrics such as NMI, ARI and purity are based on the ground-truth communities in a single level, which cannot applied to the hierarchal ground truth community. To address this and provide a fair evaluation, we

[1]https://openflights.org

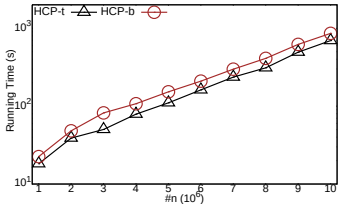
[2]https://www.openstreetmap.org/

TABLE II
RUNNING TIMES (IN SECONDS) COMPARISON FOR ALL METHODS ON EIGHT DATASETS.

Graphs	HCP-t @3	HCP-t @5	HCP-b	hLP	Paris	Spec	TeraHac@Seq
OpenFlight	**0.046**	0.078	0.048	0.05	1.44	0.50	0.83
Facebook	0.14	0.25	**0.105**	0.12	3.47	3.74	1.49
DBLP	**4.02**	6.4	6.57	6.8	260	504	6.8
Amazon	**4.31**	7.33	7.44	8.02	1037	1872	48.20
Google	**13.1**	19.7	20.56	21.7	3221	5202	142.75
Twitter	**243.5**	418.32	253.19	306.7	-	-	1142.34
LiveJournal	**75.89**	129.77	103.43	180.64	-	-	1382.24
Orkut	**180.2**	330.37	134.94	208.71	-	-	4317.83



(a) Parameter sensitivity test for the HCP-t method.



(b) Scalability test for the HCP-t and HCP-b methods.

Fig. 8. Parameter sensitivity test and scalability test for our methods.

propose a new evaluation metric based on the hierarchy similarity. For each edge $e = (u, v) \in E$, let $LCA(u, v)$ and $LCA'(u, v)$ be the number of hierarchy where the earliest community contains u and v in the original ground-truth and the clustering result respectively. The hierarchy similarity score (HSS) is given by

$$HSS = \sum_{e=(u,v)\in E} \frac{|LCA(u,v) - LCA'(u,v)|}{|E| * |\mathcal{L}|}.$$

Parameter setting: For the top-down core propagation approach HCP-t, we vary the hierarchy parameter t in $\{3, 5, 10, 15, 20\}$, and t is set to 3 by default. For other competitors, we follow the predefined parameter settings.

A. Efficiency Evaluation

Exp-1: Running time comparison of all methods. In this experiment, we compare the efficiency of all methods on the eight datasets. We report the running times of all methods in Table II. For the HCP-t method, we report its two variant by setting $t = 3$ and $t = 5$, which are denoted by HCP-t @3 and HCP-t @5 respectively. We report the running times of all methods in Table II. For the HCP-t method, we vary its input parameter t in $\{3, 5, 10, 20\}$. The mark '-' is shown if an algorithm cannot finish in 2 hours. Observed by Table II, Paris and Spec is not scalable for larger datasets, since both of them cannot finish the computation in 2 hours on the 'twitter', 'LiveJournal' and 'Orkut' datasets. Among all the methods, our HCP-t method achieve the best performance when the hierarchy $\mathcal{L}$ is set to 3. The performance drops a little when $\mathcal{L}$ grows to 5. But it's still comparative against the existing method hLP. Moreover, our HCP-t method can achieve 24X faster than the TeraHac@Seq method. The HCP-b method achieves similar performance as the HCP-t method. This experiment shows that our proposed HCP-t and HCP-b method achieve best efficiency among all existing methods.

Exp-2: Parameter sensitivity test and scalability test for our methods. For the top-down core propagation method, we vary the hierarchy parameters t in $\{3, 5, 10, 15, 20\}$, and test its efficiency sensitivity on all datasets. Fig. 8 (a) reports the running time of the top-down method on different datasets. The result shows that when the hierarchy parameter t increase linearly, the running time of HCP-t is still growing in a slow and steady rate, which demonstrates that the efficiency of HCP-t is not sensitive to the setting of t. To evaluate the scalability of our methods, we generate a series of power-law graphs using the PythonWeb Graph Generator[3]. We vary $|V|$ from 1,000,000 to 10,000,000, and $|E| = 5|V|$. Fig. 8 (b) shows the running time of our HCP-t and HCP-b methods on synthetic graphs with ascending sizes. We can observe that the running times of both methods are increasing in a stable rate, which confirms the complexity analysis in Section V and Section VI.

B. Effectiveness Evaluation

In this subsection, we compare all methods with datasets without ground-truth information under different non-ground-truth based evaluation metrics.

Exp-3: Comparison of the number of hierarchies. To evaluate the brevity of the community hierarchy structures, this experiment compares the number of levels in the hierarchical graph clustering results obtained by all methods without hierarchy parameter t, the smaller the number is, the better the hierarchical structure is. The experiment result is presented in Table III. Noted that HCP-b $^-$ is the bottom-up core propagation approach without community refinement.

[3]https://pywebgraph.sourceforge.net/

TABLE III
THE NUMBER OF HIERARCHIES OF THE HIERARCHICAL GRAPH CLUSTERING RESULTS OBTAINED BY ALL METHODS.

Graphs	HCP-b	HCP-b $^-$	hLP	Paris	Spec	TeraHac@Seq
openflight	**3**	3	4	3,096	3,096	17
facebook	**3**	3	4	4,039	4,039	21
DBLP	**4**	5	5	31,7573	31,7573	30
google	**5**	6	7	875,713	875,713	87
twitter	**4**	5	5	81,306	81,306	218
LiveJournal	**6**	8	9	4,847,571	4,847,571	873
Orkut	**8**	9	9	3,072,441	3,072,441	1,027

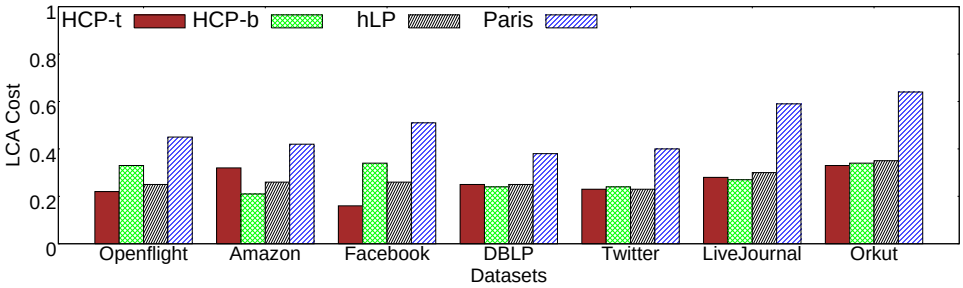
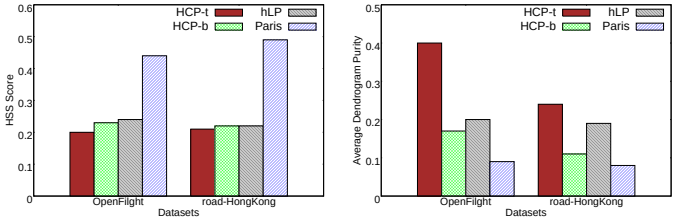


Fig. 9. LCA cost comparison for HCP-t, HCP-b, hLP and Paris on non-ground-truth graph datasets.

Since Paris and Spec share the similar merging scheme, their hierarchical graph clustering results possess number of level linear to the vertex size of the input graph, which is hard to understand and indistinguishable. Although the results obtained by TeraHac@Seq has much smaller number of levels, the hierarchies is still to much to peform analysis, (e.g., 1027 levels for the 'Orkut' dataset). hLP has similar performance to HCP-b and HCP-b $^-$, but HCP-b and HCP-b $^-$ perform better in most of the datasets, and they have equal performance to hLP on other datasets. Moreover,although HCP-b has similar performance compared to HCP-b $^-$, the HCP-b has always smaller or equal number of hierarchies, which reveals the effectiveness of the community refinement strategy proposed in Section VI whose main goal is to provide theoretical guarantee on hierarchies.

Exp-4: Least common ancestor cost evaluation. We conduct an experiment on 7 datasets without ground-truth to compare the least common ancestor cost for all methods, the smaller LCA cost represents better hierarchical clustering effectiveness. Fig. 9 shows the LCA cost for four different methods on all non-ground-truth datasets. Paris has the worst performance because there are only small difference between consecutive layers in its result. Our HCP-t and HCP-b methods have very similar performance. Compared to the hLP method, the HCP-t method can win most of the cases, which demonstrates the strong ability of our methods on guaranteeing the quality of the hierarchical graph clustering results.

Exp-5: Effectiveness evaluation on ground-truth datasets. We run an experiment on two ground-truth datasets 'OpenFlight' and 'road-HongKong' to evaluate the hierarchical similarity score of each method. As shown on Fig. 10 (a), our methods HCP-t and HCP-b have much higher HSS score than Paris, which means that the hierarchical clustering results

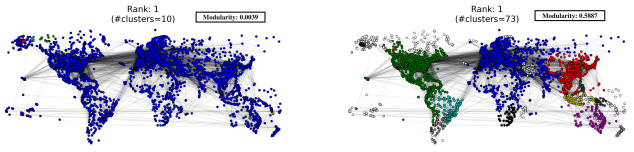


(a) HSS scores (b) Average dendrogram purity

Fig. 10. Effectiveness evaluation on ground-truth datasets.

obtained by our methods are more approaching to the ground-truth. The hLP method has very similar performance to our methods HCP-t and HCP-b. But our HCP-t method is a clear winner in both datasets, which demonstrate the superiority of our methods in terms of effectiveness. We continue to evaluate a variant of the existing dengrogram purity measure [21] called average dengrogram purity, since the traditional dendrogram measure is only based on the ground truth community in a single layer (flat layer). The average dendrogram purity uses the ground truth in each layer in the ground truth community hierarchy as input, and takes an average by dividing the number of ground truth community hierarchy layers. The larger average dendgrogram purity is, the better the effectiveness is. The result is shown in Fig. 10 (b). The result shows that our HCP-t method is still the best among all comparing methods on two ground truth datasets. HCP-b has similar performance to hLP, while paris is relatively less effective.

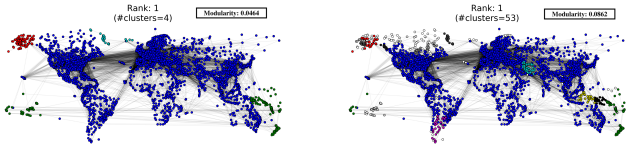
Exp-6: Case study on results comparison between HCP-t and hLP on the openflights dataset. The openflights datasets contain the flight record all over the world in a period of time. The flight records are much denser inside a particular region. And between different regions, cross-region flights are much denser for regions with closer geometric locations. The entire dataset contains a ground truth community hierarchy



(a) Clustering result of the highest layer of HCP-t.

(b) Clustering result of the second highest layer of HCP-t.

Fig. 11. Visualization results of the two highest layers obtained by our HCP-t method on the OpenFlights dataset.



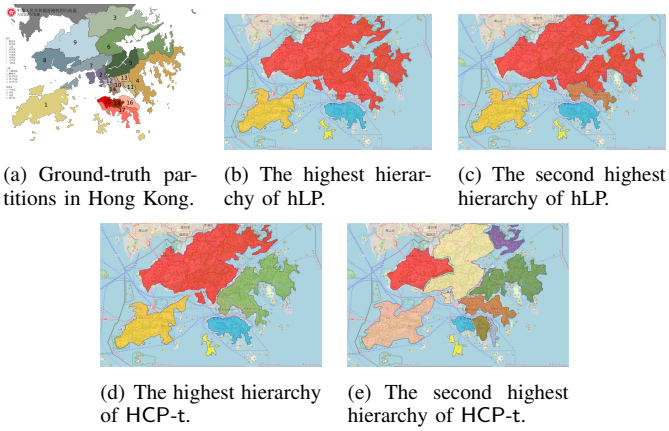
(a) Clustering result of the highest layer of hLP.

(b) Clustering result of the second highest layer hLP.

Fig. 12. Visualization results of the two highest layers obtained by the hLP method on the OpenFlights dataset.

with 2 layers, in which each vertex is associated with a country code and a content code. Fig 11 and Fig 12 visualize the clustering results of the two highest layers of our HCP-t method and the hLP method respectively. HCP-t produces a 3-layer community hierarchy with average community number difference of 84.1, while hLP produces a 4-layer result with average community number difference of 35.9. According to Fig 11 (a) and Fig 12 (a), both methods have similar clustering results on the highest layer. This is because this layer is closer to the root layer and often groups most of the communities. However, the clustering results of the second highest layers of the two methods exhibit totally different behaviors. Fig 11 (b) is the clustering result of the second highest layer of our HCP-t method. Its clustering result divides the whole world into different regions according to the location of each continent, which also follows the nature of practical flight records as mentioned before. But for the clustering result of the second highest layer of hLP as shown in Fig 12, the center part of the map is still grouped into an entire community, which has only limited change against its first layer. From the theoretical perspective, we provide the modularity values of all the clustering result on the top right side of each figure. Compared to the hLP method, the modularity of the clustering result of the second highest layer of HCP-t is 6.8 times (0.5887 against 0.0862) larger, which demonstrates that our hierarchical clustering methods is superior in terms of effectiveness from both practical and theoretical perspectives.

Exp-7: Case Study on results comparison between HCP-t and hLP on the Hong Kong road network dataset. We visualize the results of our HCP-t methods and the hLP method on the Hong Kong road network datasets. This dataset contains a ground truth community hierarchy with 2-layers, in which each vertex is associated with two community labels: district code and region code. Fig. 13 (a) shows the ground-truth administration partition of Hong Kong. Fig. 13 (b)-(e) visualize the two highest hierarchies of communities obtained by the hLP method and the HCP-t method. HCP-t produces a 3-layer community hierarchy with average community number difference of 15.7, while hLP produces a 5-layer result with average community number difference of 11.9. According to Fig. 13 (b) and (d), the two methods achieve similar clustering results in the highest hierarchy. For the results of the second highest layers in Fig. 13 (e), the entire Hong Kong is decomposed into many smaller regions by HCP-t, which is more approaching to the ground-truth in Fig. 13 (a). However, as shown in Fig. 13 (c), hLP can only decompose a small part from the center area of Hong Kong, which only makes little difference from its previous layer. Compared to the hLP method, 5 more new communities are decomposed in the second highest layer of HCP-t, which reveals the effectiveness of our information gain based hierarchy optimization technique proposed in Section V-B. This case study also shows that our HCP-t method achieve much better performance in terms of effectiveness and application on the ground-truth dataset.



(a) Ground-truth partitions in Hong Kong.

(b) The highest hierarchy of hLP.

(c) The second highest hierarchy of hLP.

(d) The highest hierarchy of HCP-t.

(e) The second highest hierarchy of HCP-t.

Fig. 13. Visualization results of the two highest layers obtained by HCP-t and hLP on the Hong Kong road network.

VIII. Conclusion

In this paper, we define an important property for hierarchical communities, and study the problem of hierarchical graph clustering. We propose two efficient core propagation algorithms from different clustering directions to solve the problem. We first propose an efficient top down core propagation method with a hierarchical parameter t for users with empirical knowledge in the input graph data. For new graph data, we further propose a parameter-free bottom up core propagation approach to obtain hierarchical graph clustering results automatically. Experiments on both ground-truth and non-ground-truth datasets demonstrate the efficiency and effectiveness of our proposed methods. Case studies on two ground-truth datasets further shows the wide applications of our methods.

IX. Acknowledgment

This work is supported by Hong Kong RGC Projects Nos. 12200424, 12201923, and C2003-23Y. Xin Huang is the corresponding author.

REFERENCES

[1] Yong-Yeol Ahn, James P Bagrow, and Sune Lehmann. Link communities reveal multiscale complexity in networks. *nature*, 466(7307):761–764, 2010.

[2] Sabeur Aridhi, Martin Brugnara, Alberto Montresor, and Yannis Velegrakis. Distributed k-core decomposition and maintenance in large dynamic graphs. In *DEBS*, pages 161–168. ACM, 2016.

[3] V. Batagelj and M. Zaversnik. An o (m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049*, 2003.

[4] Thomas Bonald, Bertrand Charpentier, Alexis Galland, and Alexandre Hollocou. Hierarchical graph clustering using node pair sampling. *arXiv preprint arXiv:1806.01664*, 2018.

[5] Francesco Bonchi, Francesco Gullo, Andreas Kaltenbrunner, and Yana Volkovich. Core decomposition of uncertain graphs. In *KDD*, pages 1316–1325. ACM, 2014.

[6] Lijun Chang and Lu Qin. *Cohesive Subgraph Computation over Large Sparse Graphs: Algorithms, Data Structures, and Programming Techniques*. Springer, 2018.

[7] James Cheng, Yiping Ke, Shumo Chu, and M. Tamer Özsu. Efficient core decomposition in massive networks. In *ICDE*, pages 51–62, 2011.

[8] James Cheng, Yiping Ke, Shumo Chu, and M. Tamer Özsu. Efficient core decomposition in massive networks. In *ICDE*, pages 51–62. IEEE Computer Society, 2011.

[9] Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

[10] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.

[11] Laxman Dhulipala, David Eisenstat, Jakub Lacki, Vahab S. Mirrokni, and Jessica Shi. Hierarchical agglomerative graph clustering in nearly-linear time. In *ICML*, volume 139, pages 2676–2686. PMLR, 2021.

[12] Laxman Dhulipala, Jakub Lacki, Jason Lee, and Vahab Mirrokni. Terahac: Hierarchical agglomerative clustering of trillion-edge graphs. *PACMMOD*, 1(3):221:1–221:27, 2023.

[13] Rundong Du, Da Kuang, Barry Drake, and Haesun Park. Hierarchical community detection via rank-2 symmetric nonnegative matrix factorization. *Computational social networks*, 4(1):1–26, 2017.

[14] Yixiang Fang, Reynold Cheng, Siqiang Luo, and Jiafeng Hu. Effective community search for large attributed graphs. *Proceedings of the VLDB Endowment*, 9(12):1233–1244, 2016.

[15] Yixiang Fang, Xin Huang, Lu Qin, Ying Zhang, Wenjie Zhang, Reynold Cheng, and Xuemin Lin. A survey of community search over big graphs. *The VLDB Journal*, pages 1–40, 2019.

[16] Yixiang Fang, Zhongran Wang, Reynold Cheng, Hongzhi Wang, and Jiafeng Hu. Effective and efficient community search over large directed graphs. *TKDE*, 31(11):2093–2107, 2018.

[17] Yixiang Fang, Yixing Yang, Wenjie Zhang, Xuemin Lin, and Xin Cao. Effective and efficient community search over large heterogeneous information networks. *Proceedings of the VLDB Endowment*, 13(6).

[18] R Franke. Chimera: Top-down model for hierarchical, overlapping and directed cluster structures in directed and weighted complex networks. *Physica A: Statistical Mechanics and its Applications*, 461:384–408, 2016.

[19] Edoardo Galimberti, Francesco Bonchi, and Francesco Gullo. Core decomposition and densest subgraph in multilayer networks. In *CIKM*, pages 1807–1816. ACM, 2017.

[20] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *PNAS*, 99(12):7821–7826, 2002.

[21] Katherine A. Heller and Zoubin Ghahramani. Bayesian hierarchical clustering. In *ICML*, volume 119, pages 297–304. ACM, 2005.

[22] Paul Jakma, Marcin Orczyk, Colin S. Perkins, and Marwan Fayed. Distributed k-core decomposition of dynamic graphs. In *StudentWorkshop@CoNEXT*, pages 39–40. ACM, 2012.

[23] Jussi M Kumpula, Mikko Kivelä, Kimmo Kaski, and Jari Saramäki. Sequential algorithm for fast clique percolation. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 78(2):026109, 2008.

[24] Matthieu Latapy. Main-memory triangle computations for very large (sparse (power-law)) graphs. *Theor. Comput. Sci.*, 407(1-3):458–473, 2008.

[25] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[26] Vincent Levorato. Core decomposition in directed networks: Kernelization and strong connectivity. In *CompleNet*, volume 549, pages 129–140, 2014.

[27] Lin Li, Kefeng Fan, Zhiyong Zhang, and Zhengmin Xia. Community detection algorithm based on local expansion k-means. *Neural network world*, (6), 2016.

[28] Rong-Hua Li, Jeffrey Xu Yu, and Rui Mao. Efficient core maintenance in large dynamic graphs. *TKDE*, 26(10):2453–2465, 2014.

[29] Qing Liu, Yifan Zhu, Minjun Zhao, Xin Huang, Jianliang Xu, and Yunjun Gao. Vac: Vertex-centric attributed community search. In *ICDE*, pages 937–948, 2020.

[30] Gang Mei, Jingzhi Tu, Lei Xiao, and Francesco Piccialli. Kcoremotif: An efficient graph clustering algorithm for large networks by exploiting k-core decomposition and motifs. *CoRR*, abs/2008.10380, 2020.

[31] Nicholas Monath, Kumar Avinava Dubey, Guru Guruganesh, Manzil Zaheer, Amr Ahmed, Andrew McCallum, Gökhan Mergen, Marc Najork, Mert Terzihan, Bryon Tjanaka, Yuan Wang, and Yuchen Wu. Scalable hierarchical agglomerative clustering. In *KDD*, pages 1245–1255. ACM, 2021.

[32] Alberto Montresor, Francesco De Pellegrini, and Daniele Miorandi. Distributed k-core decomposition. *TPDS*, 24(2):288–300, 2013.

[33] Tamás Nepusz, Andrea Petróczi, László Négyessy, and Fülöp Bazsó. Fuzzy communities and the concept of bridgeness in complex networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 77(1):016107, 2008.

[34] Mark EJ Newman. Fast algorithm for detecting community structure in networks. *Physical review E*, 69(6):066133, 2004.

[35] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[36] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. In *ISCIS*, pages 284–293. Springer, 2005.

[37] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3), 2007.

[38] Mojtaba Rezvani, Qing Wang, and Weifa Liang. Fach: Fast algorithm for detecting cohesive hierarchies of communities in large networks. In *WSDM*, pages 486–494, 2018.

[39] Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. Fast hierarchical graph clustering in linear-time. In *Companion Proceedings of the Web Conference 2020*, pages 10–12, 2020.

[40] Bilal Saoud and Abdelouahab Moussaoui. A new hierarchical method to find community structure in networks. *Physica A: Statistical Mechanics and its Applications*, 495:418–426, 2018.

[41] Ahmet Erdem Sarıyüce, Bugra Gedik, Gabriela Jacques-Silva, Kun-Lung Wu, and Ümit V. Çatalyürek. Streaming algorithms for k-core decomposition. *PVLDB*, 6(6):433–444, 2013.

[42] Radhia Toujani and Jalel Akaichi. A model based metaheuristic for hybrid hierarchical community structure in social networks. *ISi*, 1:1, 2017.

[43] Zhixiao Wang, Mengnan Hou, Guan Yuan, Jing He, Jingjing Cui, and Mingjun Zhu. Hierarchical community detection in social networks based on micro-community and minimum spanning tree. *IEICE TRANSACTIONS on Information and Systems*, 102(9):1773–1783, 2019.

[44] Dong Wen, Lu Qin, Ying Zhang, Xuemin Lin, and Jeffrey Xu Yu. I/O efficient core graph decomposition: Application to degeneracy ordering. *TKDE*, 31(1):75–90, 2019.

[45] Huanhuan Wu, James Cheng, Yi Lu, Yiping Ke, Yuzhen Huang, Da Yan, and Hejun Wu. Core decomposition in large temporal graphs. In *BigData*, pages 649–658, 2015.

[46] Wei Zhang, Feng Kong, Liming Yang, Yunfang Chen, and Mengyuan Zhang. Hierarchical community detection based on partial matrix convergence using random walks. *Tsinghua Science and Technology*, 23(1):35–46, 2018.

[47] Yikai Zhang, Jeffrey Xu Yu, Ying Zhang, and Lu Qin. A fast order-based approach for core maintenance. In *ICDE*, pages 337–348, 2017.

[48] Lihua Zhou, Kevin Lü, Peizhong Yang, Lizheng Wang, and Bing Kong. An approach for overlapping and hierarchical community detection in social networks based on coalition formation game theory. *Expert Systems with Applications*, 42(24):9634–9646, 2015.