Higher-Order Peak Decomposition

Xingyu Tan[1], Jingya Qian[2], Chen Chen[3,*], Qing Sima[1], Yanping Wu[4], Xiaoyang Wang[1], Wenjie Zhang[1]

[1] University of New South Wales, [2] Zhejiang Gongshang University, [3] University of Wollongong, [4] University of Technology Sydney
Email: xingyu.tan@unsw.edu.au

Abstract

k-peak is a well-regarded cohesive subgraph model in graph analysis. However, the k-peak model only con-siders the direct neighbors of a vertex, consequently limiting its capacity to uncover higher-order structural information of the graph.

To address this limitation, we propose a new model in this paper, named (k, h)-peak, which incorporates higher-order (h-hops) neighborhood information of vertices. Employing the (k, h)-peak model, we explore the higher-order peak decomposition problem that calculates the vertex peakness for all conceivable k values given a particular h.

To tackle this problem efficiently, we propose an advanced local computation based algorithm, which is parallelizable, and additionally, devise novel pruning strategies to mitigate unnecessary computation. Experiments as well as case studies are conducted on real-world datasets to evaluate the efficiency and effectiveness of our proposed solutions.

Preliminary

Definition1, [(k, h)-contour]. Given a graph G and two positive integers k and h, a (k, h)-contour is defined as the maximal subgraph $S \subseteq G$ that satisfies (i) $\forall v \in VS$, $degS\ (v, h) \geq k$; (ii) $\forall v \in (k', h)$-contour and $k' > k$, $v \notin S$.

Definition2, [(k, h)-peak]. Given a graph G and two positive integers k and h, (k, h)-peak is the induced subgraph of the union of (j, h)-contour, $\forall j \geq k$.

Definition3, [h-peakness]. Given a graph G and positive integer h, the h-peakness of a vertex $v \in G$, denoted by $pn(v, h)$, is interpreted as the maximum k value such that (k, h)-peak contains v.

Problem Statement. Given a graph G and an integer h, we aim to develop efficient algorithms to compute the h-peakness, i.e., $pn(v, h)$, for each vertex in G.

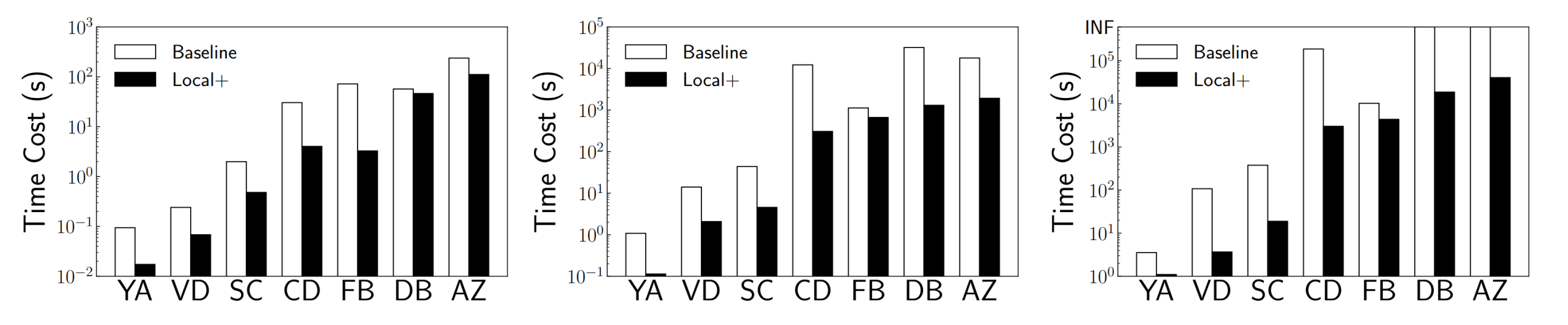
Property 1. Given a graph G and two positive integers k and h, the $(k+1, h)$-peak of G is a subgraph of its (k, h)-peak.

Property 2. Given a vertex $v \in G$, its h-peakness is equivalent to the k value such that the (k, h)-contour contains v.

Experiment

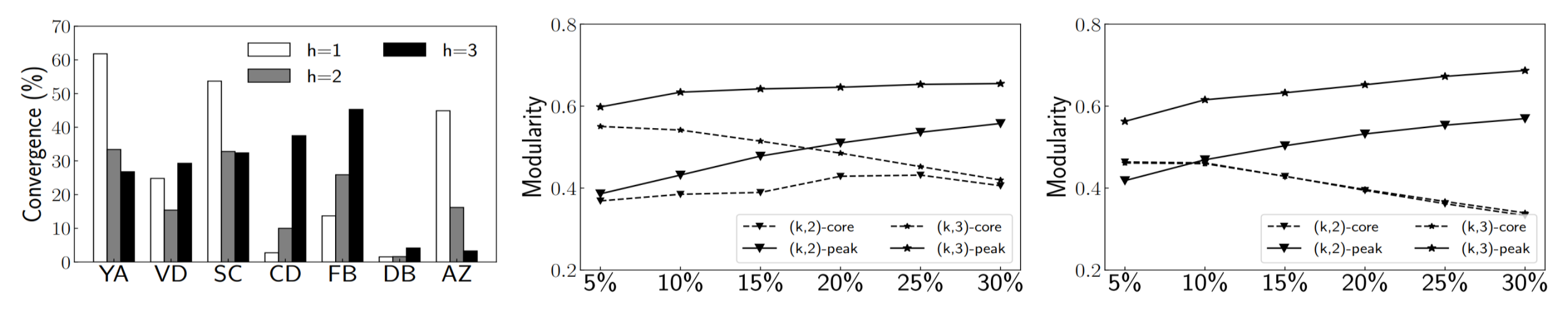
Datasets. We evaluated our algorithms on 7 real-world graphs, i.e., Yeast (YA, 2K), Vidal (VD, 6.7K), Sistercities (SC, 20.5K), Caida (CD, 53.3K), Facebook (FB, 88.2K), Douban (DB, 327.1K), and Amazon (AZ, 925.8K).

Efficiency evaluation. [Response Time (a, b, c)]



(a) $h = 1$ (b) $h = 2$ (c) $h = 3$

Effectiveness evaluation. [Early convergence rate (d), Community modularity (e, f), Number of higher-order H-index calculations (Table 1)]

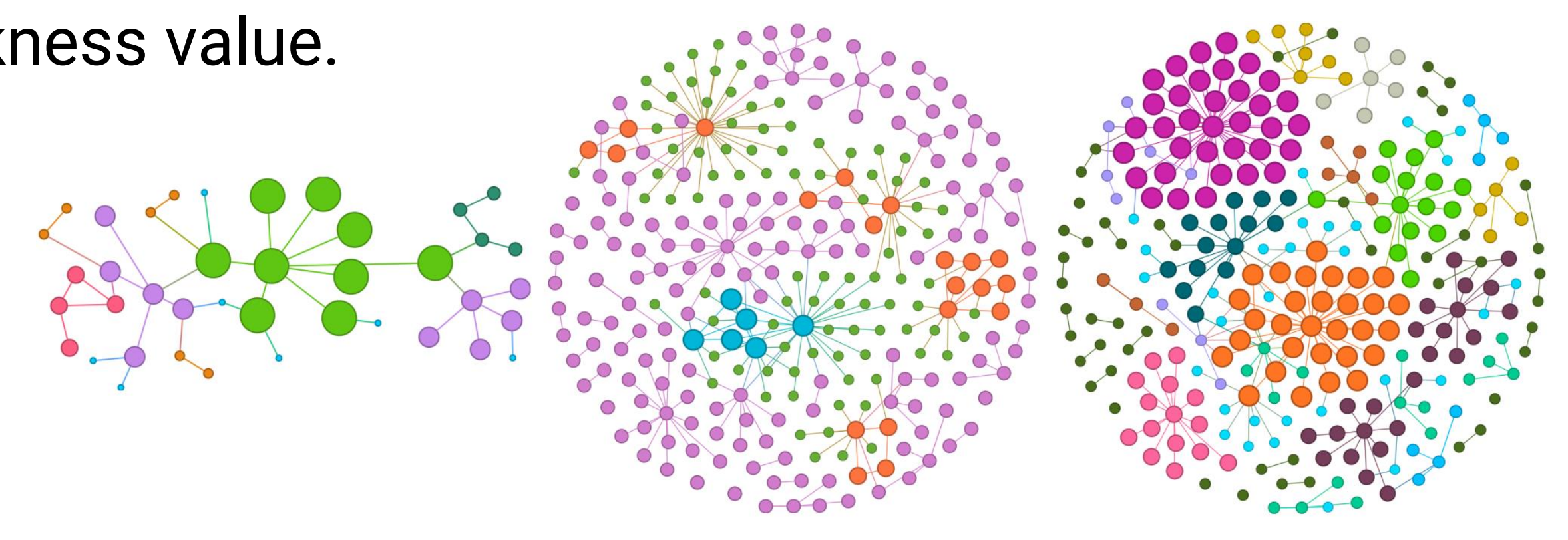


(d) Convergence (e) CD (f) DB

Table 1: Pruning strategy evaluation

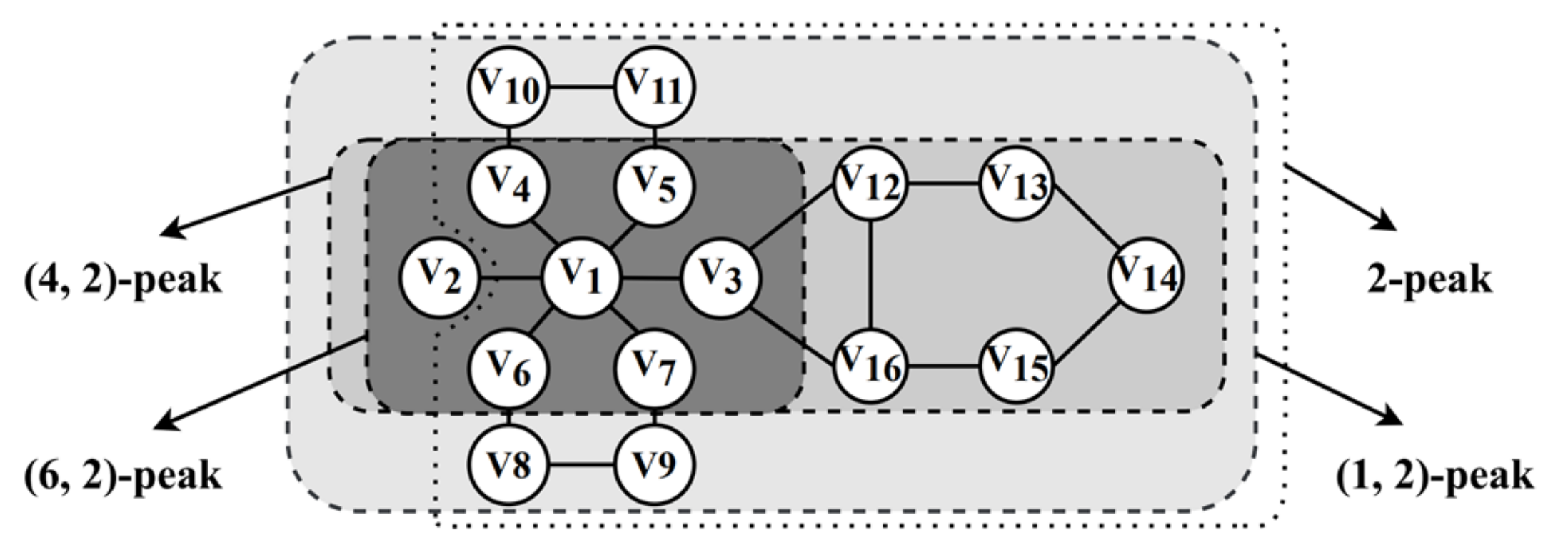
	Yeast			Vidal			Sistercitis			Caida			Facebook			Douban			Amazon		
h	**Local**	**Local+**	**%**	**Local**	**Local+**	**%**	**Local**	**Local+**	**%**	**Local**	**Local+**	**%**	**Local**	**Local+**	**%**	**Local**	**Local+**	**%**	**Local**	**Local+**	**%**
h = 1	0.29K	0.21K	29%	6K	4K	31%	18K	11K	37%	66K	27K	58%	263K	142K	46%	892K	248K	72%	473K	366K	23%
h = 2	24K	14K	42%	449K	150K	67%	1.5M	369K	75%	6.4M	3.7M	42%	4.0M	2.1M	46%	148M	32.9M	78%	22.5M	10M	54%
h = 3	216K	95K	56%	663K	200K	70%	4M	1M	76%	1.2M	0.7M	41%	13.1M	6.9M	48%	391M	96.8M	75%	641M	175M	73%

Case Study. Results of higher-order peak decomposition on Disease Spread Network (HIV) and Gene Fusion Network (Gene). Vertices with the same color have identical h-peakness, and the size of each vertex is proportional to its h-peakness value.



(a) HIV, $h = 2$ (b) Gene, $h = 1$ (c) Gene, $h = 2$

Example



The (k, h)-peak model is advantageous in revealing fine-grained structural information based on the introduction of higher-order neighborhoods. In Figure, suppose $h = 2$. The traditional peak model would treat the whole graph as a 2-peak, excluding the $v2$. Through higher-order peak decomposition, the graph can be dissected into (6, 2)-peak, (4, 2)-peak, and (1, 2)-peak components, i.e., with fine-grained structures.

Algorithms

Baseline: Based on the following lemma, we present the baseline algorithm by iteratively peeling off the largest (k, h)-core from graph.

Lemma 1. Given a graph G and an integer h, for the maximum k value of (k, h)-core existing in G, i.e., k equals to the largest h-coreness, the (k, h)-contour of G is equivalent to the (k, h)-core.

Local Algorithm: Motivated by "vertex's H-index would finally converge to its coreness". we propose a **local computation based strategy** by leveraging the local neighbour information to iteratively update and calculate the h-coreness of a vertex until converge. And it is highly parallelizable in an asynchronous manner.

Local+ Algorithm: By integrating the **early termination** and **pruning strategy** with the local computation based framework, we propose the Local+ approach, whose details are shown in Algorithm1.

Lemma2, [Early convergence]. Given (k, h)-shell of G, let S be the (k, h)-core computed on (k, h)-shell. Then, the h-peakness of all the vertices in S is k.

Lemma3, [Pruning Strategy]. Given two vertex $v, u \in G$, suppose $v \in NG\ (u, h)$ and the h-coreness of u has changed in the current iteration. Let $pre\_cn(u, h)$ and $new\_cn(u, h)$ be the original and new h-coreness of u, respectively. Then, the following two cases are invalid transfers: $i)$ $new\_cn(u, h) \geq cn(v, h)$, and $ii)$ $pre\_cn(u, h) < cn(v, h)$.

Algorithm 1: Local+ (k, h)-Peak Decomposition Algorithm

```
Input: The original graph $G(V, E)$, distance $h$
Output: the $h$-peakness for each vertex $v \in V$
compute $cn(v, h)$ for each $v \in V$ via $(k, h)$-core decomposition;
compute $h$-peakness for vertices satisfying Lemma 2;
while $G \neq \emptyset$ do
    $k_{max} \leftarrow max\{cn(v, h) : v \in V\}$;
    for each $v \in G$ with $cn(v, h) = k_{max}$ do
        $pn(v, h) \leftarrow k_{max}$;
        for each $u \in N_G(v, h)$ do
            if $visited(u) = false \wedge cn(u, h) \neq k_{max}$ then
                $Q.push(u)$; $visited(u) \leftarrow true$;
        $G \leftarrow G \setminus v$;
    for each $v \in Q$ in parallel do
        $new\_cn(v, h) \leftarrow$computeH$(v, h)$;
        if $new\_cn(v, h) \neq cn(v, h)$ then
            $cn(v, h) \leftarrow new\_cn(v, h)$; UpdateNB$(v, G)$;
Procedure UpdateNB$(v, G)$
for each $u \in N_G(v, h)$ do
    if $new\_cn(v, h) \geq cn(u, h)$ then continue;
    if $pre\_cn(v, h) < cn(u, h)$ then continue;
    $new\_cn(u) \leftarrow$computeH$(v, h)$;
    if $new\_cn(u) \neq cn(u)$ then
        $pre\_cn(u) \leftarrow cn(u)$; $cn(u) \leftarrow new\_cn(u)$;
        UpdateNB$(u, G)$;
$\mathcal{P} \leftarrow \{pn(v, h) : v \in V\}$;
return $\mathcal{P}$;
```