Neighborhood based hypergraph core decomposition

Naheed Anjum Arafat*, Arijit Khan+, Arpit Kumar Rai#, Bishwamittra Ghosh$

*Nanyang Technological University (Singapore),
+Aalborg University (Denmark),
#IIT Kanpur (India),
$A*STAR (Singapore)

PVLDB 2023

Introduction

Hypergraph

A **hypergraph** (V, E) consists of a set of nodes V and a collection of subsets of nodes E called *hyperedges*. Unlike edges in a graph, hyperedge may contain more than 2 nodes.
Examples: co-authorship in papers, event-participant relations in meet-ups, etc.

Neighbors

Pair of nodes that co-occur in a hyperedge are neighbours.

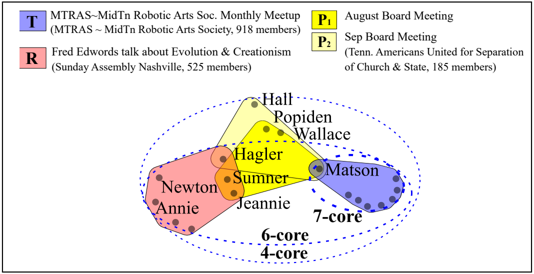


Figure 1: The set of events $H = \{T, R, P_1, P_2\}$ forms a hypergraph. *Annie* and *Newton* are neighbors. *Newton* has 6 neighbours.

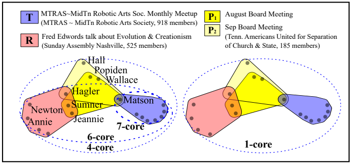
Neighborhood based core decomposition

Neighborhood-based core decomposition

Decomposition of a hypergraph into nested, maximal subhypergraphs/cores such that all nodes in the k-core have at least k neighbors in that subhypergraph.
Examples: 6-core => $\{T, R\}$, 7-core => $\{T\}$

Applications

Intervening propagation of contagions, finding influential nodes for viral marketing campaigns, densest subhypergraph extraction etc.



(left) Neighborhood-based and (right) degree-based core decomposition of a hypergraph H

Motivation

Limitations of existing methods.

Hypergraph Degree-based decomposition may not be informative

Reduced Hypergraph Reducing to Clique graph and bipartite graph and then applying graph core-decompositions produces non-intuitive results.

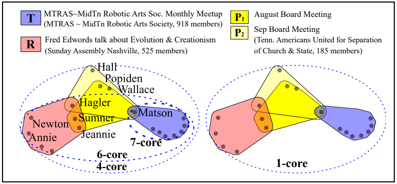


Figure 2: (left) Neighborhood-based and (right) degree-based core decomposition of a hypergraph H

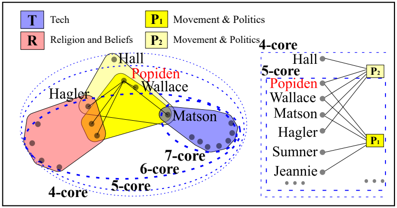


Figure 3: Alternative decompositions (a) Core decomposition of clique graph of H and (b) Dist-2 core decomposition of the bipartite graph of H. **Non-intuitiveness:** Similar events (P_1 and P_2) in different cores.

Challenges

Peeling paradigm In the classic peeling algorithm for graph, a node removal reduces its neighbors' degree by 1 (Linear time algorithm). However, in a neighborhood-based hypergraph core decomposition, its neighboring nodes # neighbors may reduce by more than 1 (Polynomial time).

Local algorithm paradigm Graph *h*-index reports incorrect neighborhood-based core.

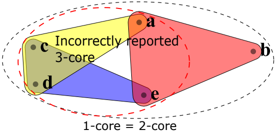


Figure 4: For any $n > 1$, the h-index of node a never reduces from $h_a^{(1)} = \mathcal{H}(2, 3, 3, 4) = 3$ to its correct core-number 2. Because a will always have at least 3 neighbors (c, d, and e) whose h-indices are at least 3. An incorrect 3-core reported.

Problem Statement

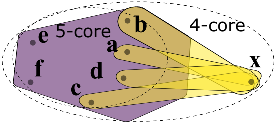
How to correctly and efficiently compute neighborhood-based hypergraph cores.

Naive Peeling algorithm: **Peel**

1. At each iteration $k \in \{1, 2, \cdots, |V|\}$,
   1. Remove the node with # neighbors $\leq k$.
   2. Report k as the core-number of the removed node.
   3. Recompute the #neighbors of neighboring nodes.
2. Complexity: $\mathcal{O}(|V| . d_{nbr} . (d_{nbr} + d_{hpe})$, here d_{nbr} (d_{hpe}) is the #neighbor (degree) of the node with largest #neighbors (degree).

Can we do better?.

Delay # neighbors recomputation of nodes with core-number $> k$ based on lower-bound.



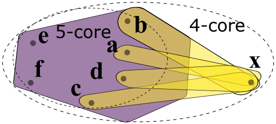
Node b's #neighbors is recomputed twice: (1) when x is peeled and later (2) when e is peeled. Can we delay the recomputation until e is peeled?

Better Peeling algorithm: **E-Peel**

1. Compute the core-number lower bound for all nodes.
2. At each iteration $k \in \{1, 2, \cdots, |V|\}$,
   1. Remove the node with # neighbors $\leq k$.
   2. Report k as the core-number of the removed node.
   3. Recompute the #neighbors of a neighboring node v only if $k >= LB(v)$.

$$LB(v) = \max\left(|e_m(v)| - 1, \min_{u \in V} |N(u)|\right)$$

Here $e_m(v)$ is the maximal cardinality hyperedge containing v



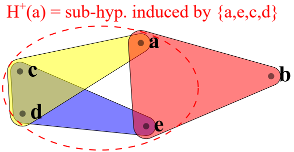
Node b's #neighbors computation is delayed until e is peeled.

Best algorithm: Local core and optimisations

Input: Hypergraph $H = (V, E)$
Output: Core-number $c(v)$ for each node $v \in V$

for all $v \in V$ **do**
  $\hat{h}_v^{(0)} = h_v^{(0)} \leftarrow |N(v)|$.
for all $n = 1, 2, \ldots, \infty$ **do**
  for all $v \in V$ **do**
    $h_v^{(n)} \leftarrow \min\left(\mathcal{H}(\{\hat{h}_u^{(n-1)} : u \in N(v)\}), \hat{h}_v^{(n-1)}\right)$
  for all $v \in V$ **do**
    $c(v) \leftarrow \hat{h}_v^{(n)} \leftarrow$ **Core-correction** $(v,\ h_v^{(n)},\ H)$
  if $\forall v, \hat{h}_v^{(n)} == h_v^{(n)}$ **then**
    Terminate Loop
Return c

Core correction:



$h_a = h_e = h_c = h_d = 3.\ \ h_b = 2$
$H^{+}(a) = H[\{u: h_u >= h_a\}]$

Reduce h-index h_a **by 1 until the #neighbors of** a **in** $H^{+}(a) \geq h_a$**: Node** a**'s corrected h-index = 2.**

Hypergraph h-index of order n

The Hypergraph h-index of order n for node v, denoted as $\hat{h}_v^{(n)}$, is defined for any natural number $n \in \mathbb{N}$ by the following recurrence relation:

$$\hat{h}_v^{(n)} = \begin{cases} |N(v)| & n = 0 \\ h_v^{(n)} & n > 0 \wedge LCCSAT(h_v^{(n)}) \\ \max\{k \mid k < h_v^{(n)} \wedge LCCSAT(k)\} & n > 0 \wedge \neg LCCSAT(h_v^{(n)})\textbf{(Core-correction)} \end{cases} \tag{1}$$

Local core: Theoretical Gurantees

Hypergraph h-index has a limit

For any node $v \in V$ of a hypergraph $H = (V, E)$, the two sequences $(h_v^{(n)})$ and $(\hat{h}_v^{(n)})$ have the same limit: $\lim_{n\to\infty} h_v^{(n)} = \lim_{n\to\infty} \hat{h}_v^{(n)}$.

The limiting value is the core-number

If the local coreness-constraint is satisfied for all nodes $v \in V$ at the terminal iteration, the corrected h-index at the terminal iteration $\hat{h}_v^{(\infty)}$ satisfies: $\hat{h}_v^{(\infty)} = c(v)$.

Convergence time guarantee

Given a node $v \in N_i$ in a hypergraph H, it holds that $\forall n \geq i$, $\hat{h}_v^{(n)} = c(v)$. Here N_i is the i-th *neighborhood hierarchy*, which contains the set of nodes that have the minimum number of neighbors in $H[V']$, where $V' = V \setminus \cup_{0 \leq j < i} N_j$

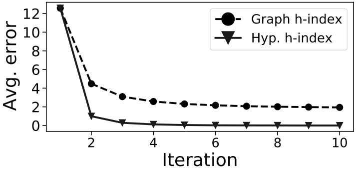
Optimisations and parallelisation of Local core

- **Optimisations:** We have proposed 4 optimisations to make **Local-core** more efficient.
- **Parallisation:** We have proposed **Local-core(p)**, a shared-memory, data parallel programming adaptation of Local-core.
- **Generalised core model**: We have proposed a generalised hypergraph core model *(neighborhood, degree)-core* that simultaneously considers degree constraint and neighborhood constraint.

Datasets

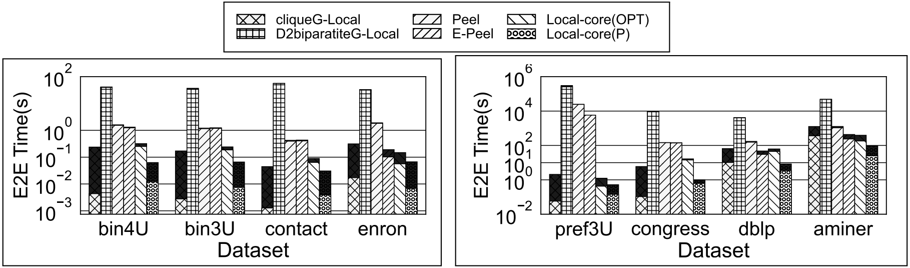
Table 1: Datasets: $|V|$ #nodes, $|E|$ #hyperedges, $d(v)$ (mean) degree of a node, $|e|$ (mean) cardinality of a hyperedges, $|N(v)|$ (mean) #neighbors per node

	hypergraph	$\|V\|$	$\|E\|$	$d(v)$	$\|e\|$	$\|N(v)\|$
Syn.	*bin4U*	500	12424	99.4±8.5	4±0	225.3±15.5
	bin3U	500	16590	99.5±8	3±0	164.1±11.6
	pref3U	125329	250000	5.9±915.9	3±0	4.5±412.4
Real	*enron*	4423	5734	6.8±32	5.2±5	25.3±44
	contact	242	12704	127±55.2	2.4±0.5	68.7±26.6
	congress	1718	83105	426.2±475.8	8.8±6.8	494.7±248.6
	dblp	1836596	2170260	4±11.6	3.4±1.8	9±21.4
	aminer	27850748	17120546	2.3±5	3.7±2.6	8.4±24.1



Importance of Hypergraph h-index: Average error of hyp. and graph *h*-index on *Enron*

Efficiency evaluation



(a)

(b)

Figure 5: (a)-(b) End-to-end (E2E) running time of our algorithms: **Peel**, **E-Peel**, **Local-core(OPT)**, **Local-core(P)** with 64 Threads vs. those of baselines: **Clique-Graph-Local** and **Distance-2 Bipartite-Graph-Local**. End-to-end (E2E) running time = data structure initialization time (shaded with dark-black on top of each bar) + algorithm's execution time.

Our OpenMP parallel implementation **Local-core(P)** decomposes *aminer* hypergraph with 27M nodes, 17M hyperedges in 91 seconds.

Application 1: Densest subgraph discovery

- A new notion of densenset sub-hypergraph.

Volume-densest subhypergraph

The **volume-densest subhypergraph** is a subhypergraph which has the largest volume-density among all subhypergraphs. The **volume-density** $\rho^N[S]$ of a subset $S \subseteq V$ of nodes in a hypergraph $\rho^N[S] = \frac{\sum_{u \in S} |N_S(u)|}{|S|}$.

- **Greedy approximation algorithm** for volume-densest subhyp. recovery is $(d_{pair}(d_{card} - 2) + 2)$-approximate, where hyperedge-cardinality and node-pair co-occurrence (# hyperedges containing that pair) are at most d_{card} and d_{pair}, resp.

Case study: Nashville Meetup Dataset

- The degree-densest subhyp. contains casual, frequent gatherings from only one socializing group. (**Not informative**)
- The degree-densest subgraph of the clique graph captures technical events arranged by diverse, yet niche activity groups (e.g. 5 participants on avg.) (**Informative**)
- The volume-densest subhyp. captures technical events arranged by diverse and vibrant activity groups (78 participants on avg.) (**More informative**)

Application 2: Influence spreading and intervention

Initially, all nodes except one called a *seed*- are at the *susceptible* state. The seed node is initially at the *infectious* state. At each time step, each infected node infects its susceptible neighbors with probability β and then becomes *immunized*. Once a node is immunized, it is never re-infected.

1. Inner-cores produced by our decomposition contain influential spreaders.
2. Our decomposition produces the best order of important nodes for deleting a limited number of them while causing the maximum intervention in spreading.

Conclusion

- We introduced neighborhood-cohesive core decomposition of hypergraphs.
- We proposed efficient algorithms for hypergraph core decomposition.
- Applications:
  - **Densest subhypergraph extraction.** Case studies show that our novel volume-densest subhypergraphs capture differently important meetup events, compared to both degree and clique graph decomposition-based densest subhypergraphs
  - **Diffusion intervention.** Our proposed decomposition is more effective than the degree and clique graph-based decompositions in intervening diffusion.

Future work.

Efficient algorithms for the new hypergraph-core model, *(neighborhood, degree)-core*