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)

Hypergraphs

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, a 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 neighbors.

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.

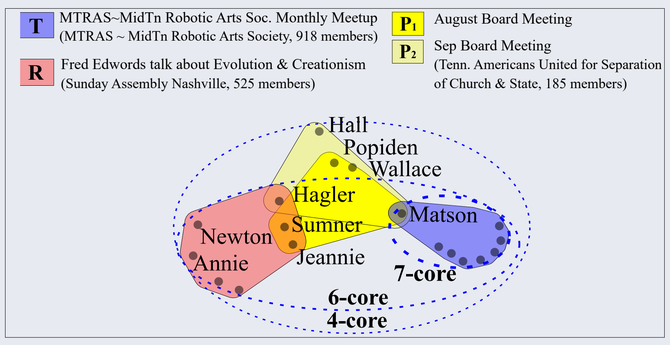


Figure 1:The set of events $H = \{T, R, P_1, P_2\}$ forms a hypergraph. *Annie* and *Newton* are neighbors. *Newton* has 6 neighbors. 6-core => $\{T, R\}$, 7-core => $\{T\}$

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

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.

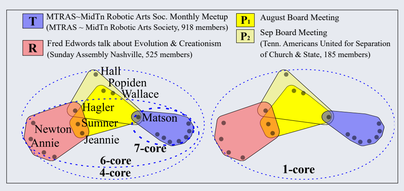


Figure 2:(left) Neighborhood-based and (right) degree-based core decomposition of a hypergraph H

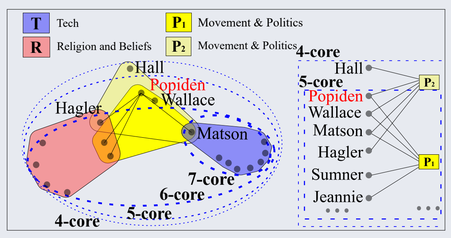


Figure 3:Alternative decompositions (a) Core decomposition of clique graph of H and (b) Dist-2 core decomposition of the bipartite graph of H.

Non-intuitiveness: Similar events (P_1 and P_2) in different cores.

Our Contributions

- A **novel core decomposition** of hypergraphs.
- **Algorithms**:
  - Naïve algorithm: Peel
  - Efficient peeling: E-Peel
  - Local algorithms: Loca-core, Local-core with optimisations and parallelization.
  - Generalisation: (neighborhood,degree)-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.

Problem Statement

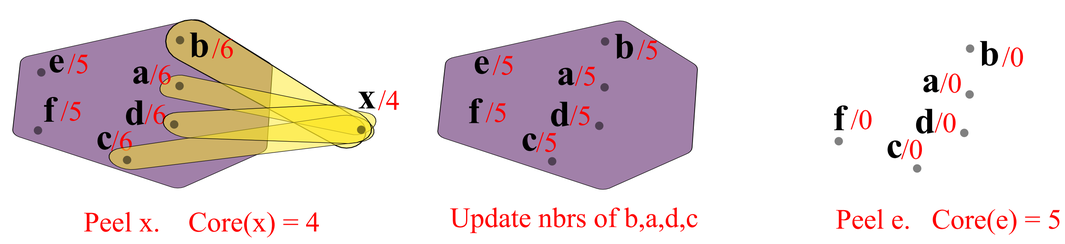
How to correctly and efficiently compute neighborhood based hypergraph cores?

Algorithms

Peel

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.



Can we do better?.

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

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

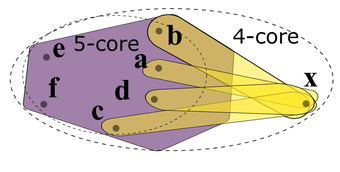


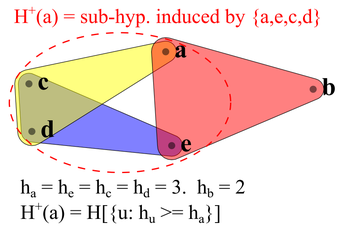
Figure 4:Node b's #neighbors computation is delayed until e is peeled.

Local-core

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:



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.

- **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.

Efficiency comparison

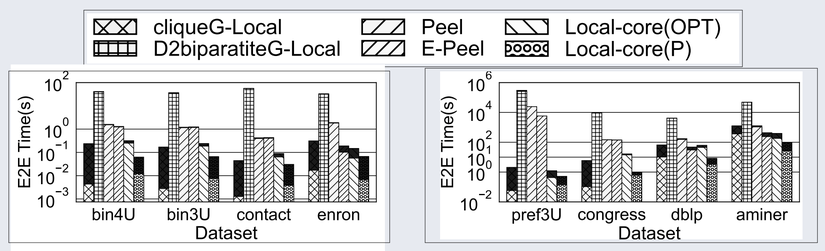


Figure 5:End-to-end (E2E) running time of our algorithms: **Peel**, **E-Peel**, **Local-core(OPT)**, **Local-core(P)** with 64 Threads vs. those of baselines: **Clique-Graph-Local** and **Distance-2 Bipartite-Graph-Local**

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