Efficient Progressive Minimum k-Core Search

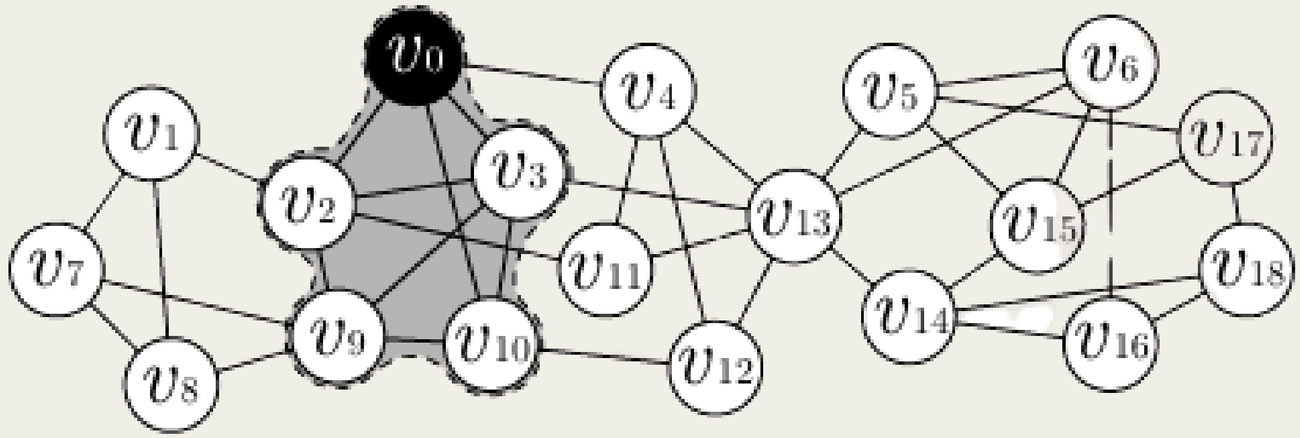
Authors: Conggai Li , Fan Zhang, Ying Zhang, Lu Qin, Wenjie Zhang and Xuemin Lin

COP5725 Database Systems

Team: Deepak Kukkapalli [vk23p]
Karthik Reddy Vemireddy [kv23b]
Muqeet Mohsin Shaik [m23ch]

What is a K-Core?

A k-core is a subgraph where each node has at least k connections to other nodes in the same subgraph. We are trying to find the smallest k-core subgraph (minimum k-core) that includes certain important nodes (called query vertices).



Example: where k =3

Graph **G=(V,E)** where:

- **V** is the set of vertices (nodes).
- **E** is the set of edges (connections) between the vertices.
- **n = |V|** is the number of vertices in the graph.
- **m = |E|** is the number of edges in the graph.

Problem and Motivation

- Finding this smallest k-core is very difficult because it's an NP-hard problem, meaning there's no easy, fast way to always find the perfect solution.

- The search space is huge, making it impractical to find the exact minimum k-core by brute force. Existing methods mostly focus on greedy approaches, which may not find the best possible solution.

Motivation [Why focus on the smallest k-core?]:

Social Networks: If a company wants to recommend a product to a group of users, a smaller, more relevant group (instead of a huge one) makes more sense.
Biological Networks: When studying proteins that interact, scientists reported small, closely related groups share the same function.
Neural Networks: In brain research, activating a small k-core of neurons can be more efficient than targeting a large group.

EXISTING SOLUTIONS FOR MINIMUM K-CORE SEARCH

Global Search (Shrink Strategy)

1. Initialization: Compute the maximal k-core of the graph.
2. Pruning: Iteratively remove non-query vertices while ensuring the remaining subgraph retains the k-core property.
3. Goal: Minimize subgraph size while retaining the query vertex.

Local Search (Expansion Strategy)

1. Initialization: Start with the query vertex q as subgraph P.
2. Expansion: Maintain a candidate set C (neighbors of P not yet added). Iteratively add the highest-scoring vertex u from C using S-Greedy: Score(u) = $p^+(u) - p^-(u)$
    - $p^+(u)$: Fixes low-degree nodes in P (neighbors of u with degree < k).
    - $p^-(u)$: Additional nodes required to ensure u has ≥ k neighbors.
    - Stop when P becomes a valid k-core.
3. Time Complexity: $O(s(d_{max} + \log n))$, where s = output size, d_{max} = max degree.

LIMITATIONS OF EXISTING APPROACHES

1. *Not optimal:* The greedy approach does not guarantee that the final subgraph is the smallest possible k-core.
2. *Quality gap:* Empirical studies show that the subgraphs produced are still much larger than the true minimum k-core.

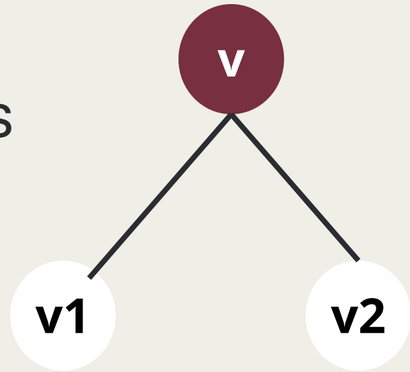
Need for a New Approach:

Since existing methods lack control over result quality and do not guarantee an optimal or near optimal subgraph, so we'd need an algorithm that balances both quality and efficiency.

PROGRESSIVE SEARCH ALGORITHM (PSA)

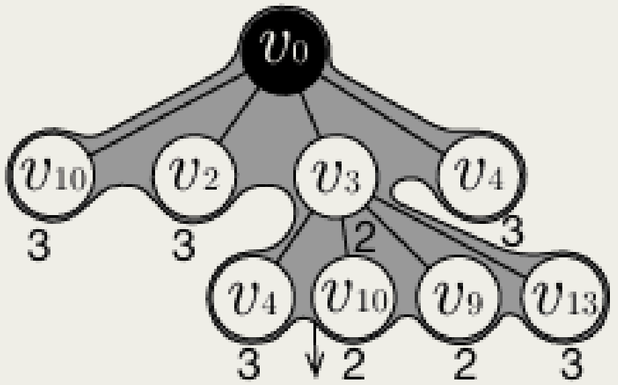
PSA (Progressive Search Algorithm) is a method used to find an approximate minimum k-core in a graph. It does this by combining Lower Bound Search and Upper Bound Search to efficiently explore and refine a set of nodes.

- The algorithm starts at a given query node v.
- It checks if v belongs to the minimal k-core. If not, the process stops.
- Otherwise, the algorithm begins constructing a search tree, where each node represents a possible choice of expanding the subgraph.



LOWER BOUND SEARCH

- This approach expands the most promising nodes first, ensuring that the subgraph remains strongly connected.
- At each step, the algorithm selects a node with a high degree (many connections) and grows outward.
- If a new node has a stronger connection than the previously chosen one, it is prioritized to build a solid foundation.

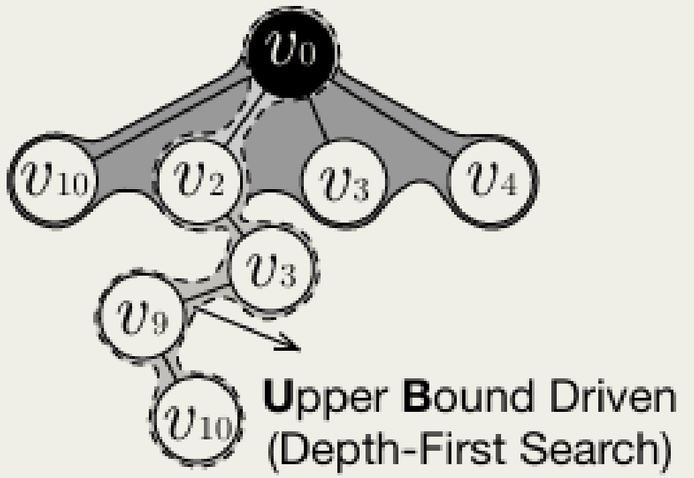


Lower **B**ound Driven
(Best-First Search)

Think of it like forming a close-knit friend group in a social network — first, you pick the people who have the most connections to the existing members.

UPPER BOUND SEARCH

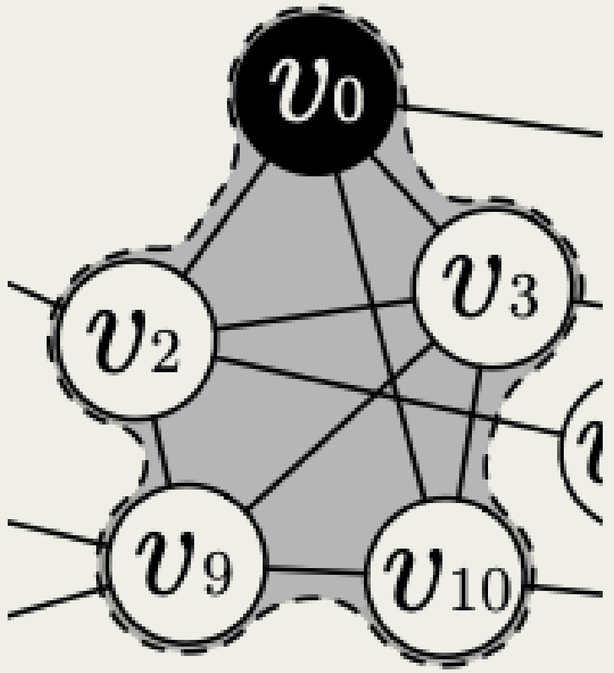
- This approach tests deeper paths quickly, attempting to find a complete solution earlier.
- If a path leads to a valid k-core, the algorithm updates its best-known result.
- If a path appears weak (not forming a strong k-core), it is discarded early, avoiding unnecessary exploration.



This is like testing a small but tight group of friends to see if they form a strong enough core before looking at wider connections.

COMBINING LOWER AND UPPER BOUNDS (PSA ALGORITHM)

- The PSA algorithm runs both Lower Bound Search and Upper Bound Search together.
- If a newly discovered subgraph is better than the current best solution, the algorithm updates its selection.
- The process continues until the best possible k-core is found while ensuring the approximation ratio is met.



Imagine trying to find the best-connected social circle — one approach expands steadily, while another checks deeper paths to find a good fit quickly. Together, they ensure an optimal.

KEY RESULTS

PSA vs. State-of-the-Art: Superior Effectiveness

- Result Size:
  - PSA produces 2–10x smaller k-cores than S-Greedy.
  - Example: Yeast dataset: 52 nodes (PSA) vs. 200+ (S-Greedy).

- Engagement & Structure:
  - 49% engagement (vs. ≤43% for others) with lower diameter (3.39 vs. 5–14).
- Takeaway: PSA ensures minimal, actionable communities with higher relevance.

Metrics	diameter	degree	density	CC	avg. size	Engage
k-Core	8.12	24.45	0.001	0.32	23196.29	35%
k-Truss	4.61	6.92	0.63	0.33	13020.85	43%
k-Ecc	7.62	25.18	0.02	0.33	22561.75	36%
k-Clique	5.18	15.49	0.38	0.63	15775.21	42%
Graph Clustering	13.87	8.59	0.11	0.32	81232.53	30%
Min k-Core	3.39	10.09	0.37	0.57	52.02	49%

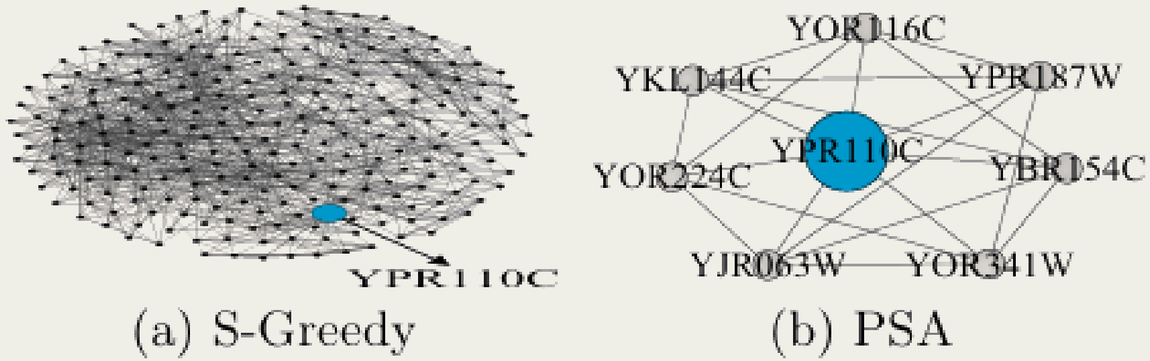
On Gowalla Dataset

KEY RESULTS

Real-World Precision: Case Study on Yeast

- Biological Networks:
  - PSA’s 7-node subgraph: All proteins share functionality with the query.
  - S-Greedy’s 254-node result: Only 12.6% functional relevance.

- Implications:
  - Enables precise protein-function prediction.
  - Reduces verification costs in recommendation systems.



(a) S-Greedy (b) PSA

KEY RESULTS

Unmatched Efficiency & Scalability

- Speed:
  - 2–5x faster than baselines.
  - Processes billion-edge graphs (e.g., Web base) in 1.5 hours.

- Technical Breakthroughs:
  - Tight bounds: lower and upper bounds reduce search space which ensures minimal k-cores.

For Wiki dataset

CONCLUSION

- **PSA Delivers:**
  - Quality: Near-optimal k-cores with provable guarantees.
  - Speed: 2–5x faster than competitors.
  - Versatility: Effective for hubs, non-hubs, and multi-query scenarios.
- **Real-World Value:** Enables actionable insights in social networks, bioinformatics, and recommendation systems.