Scalable K-Truss Implementation in Arkouda

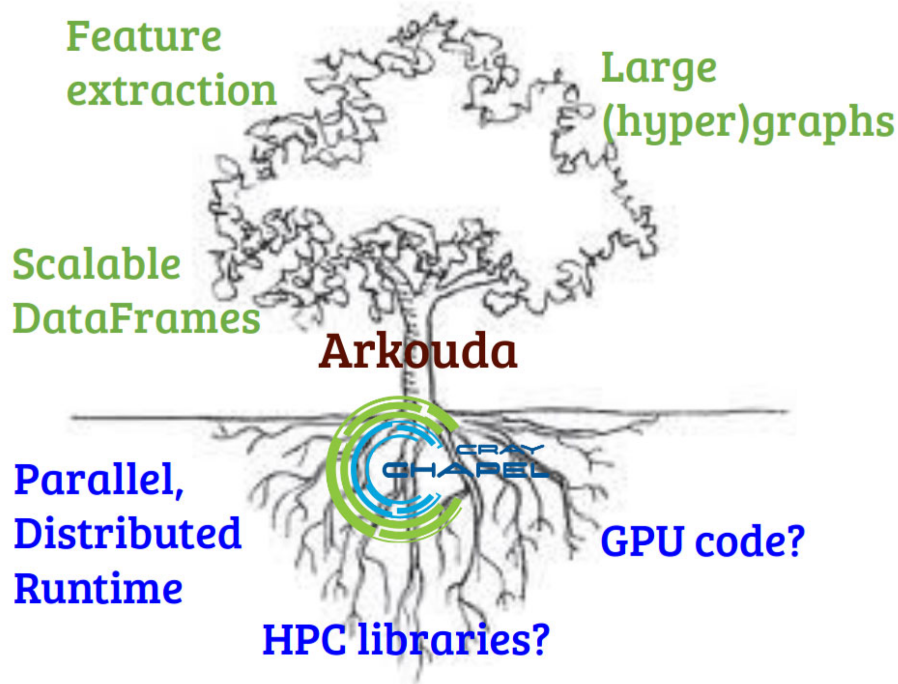
Joseph Patchett (Presenter)
Zhihui Du, Oliver Alvarado Rodriguez,
David Bader

2022

This research is supported by NSF grant CCF-2109988

The Arkouda Framework [Reus, Merrill 2019]

- Interactive Python frontend
  - Abstracts HPC away with pythonic tools
  - Provides interactive manipulation of massive graph datasets
- ZMQ socket for communication between interfaces
- Chapel backend for handling storage and HPC tools
  - Open-source framework originally developed by the Department of Defense



[Reus 2020]

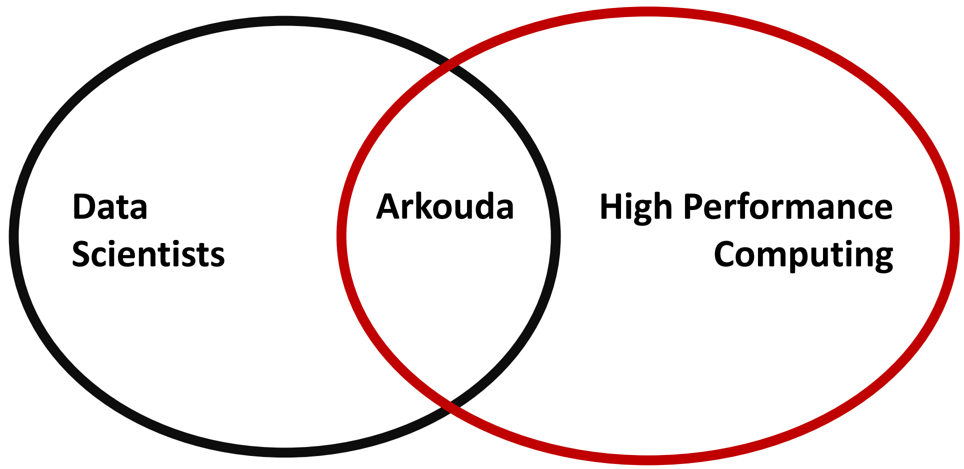
New Jersey Institute of Technology

Why use Arkouda?

- Growth of graph datasets have left personal computers in the dust

- To illustrate this growth:
  - In 2005, 5% of adults in the US used social media.
  - In 2020 72% of adults used social media, nearly a 1500% increase!
- Arkouda has applications in cybersecurity, network logs, corporate data, and many other datasets

- Arkouda abstracts HPC away from data scientists to allow them to focus on the data itself



NJIT New Jersey Institute of Technology

Our Contributions

- What we have done:
  - Exploit our previous work to accelerate the development of k-truss method quickly (high developing performance).
  - Design an optimized multi-locale (distributed) parallel k-truss algorithm that utilizes minimal degree for decomposition and peeling.
  - Developed a k-truss peeling method, a method to find max-k, and a complete truss decomposition method.

New Jersey Institute of Technology

The k-Truss Decomposition

- k-Truss Decomposition is a subgraph where all edges are incident to k-2 triangles
  - Other edges are removed.
- The decomposition can be applied to reduce large networks into more meaningful subgraphs

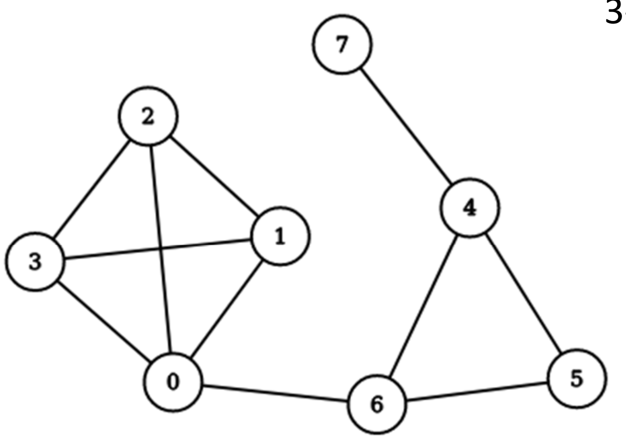
- k-Truss shows the connectedness of a graph.
  - Applications in social media analysis and graph mining.

Why k-Truss?

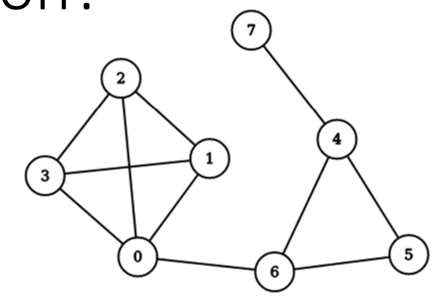
- In Social Media, we can use k-Truss for Exploratory Data Analysis (EDA)
- By searching for a certain k-value, we can find people who are highly connected with others
- Alternatively, we can isolate communities of social media spam bots that form highly interactive communities within themselves

NJIT New Jersey Institute of Technology

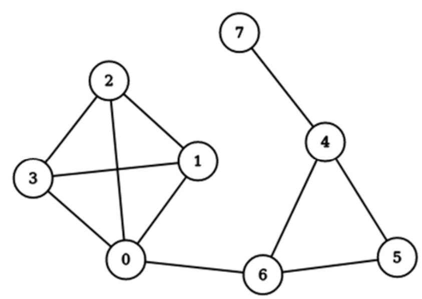
Can you find the k-Truss Decomposition?



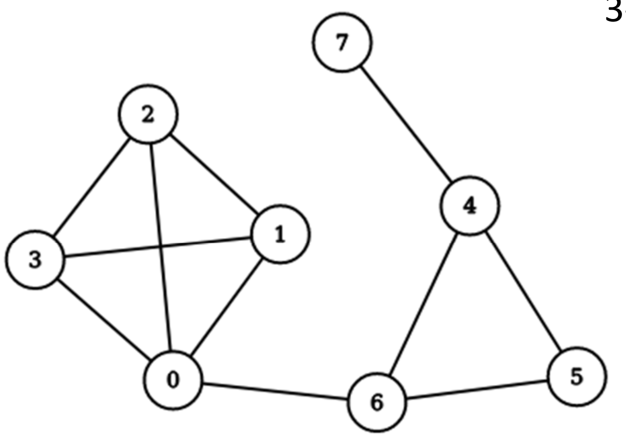
3-Truss Decomposition



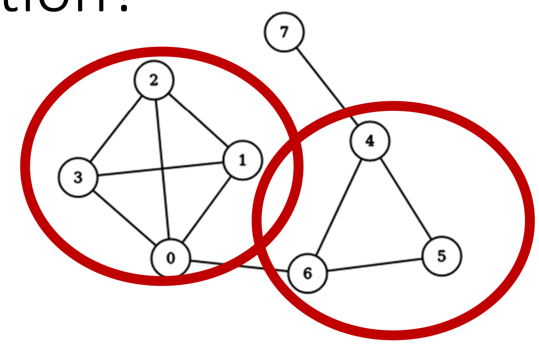
4-Truss Decomposition



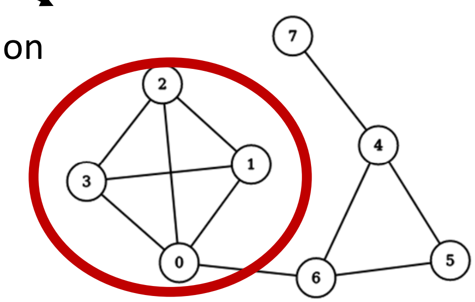
Can you find the K-Truss Decomposition?



3-Truss Decomposition

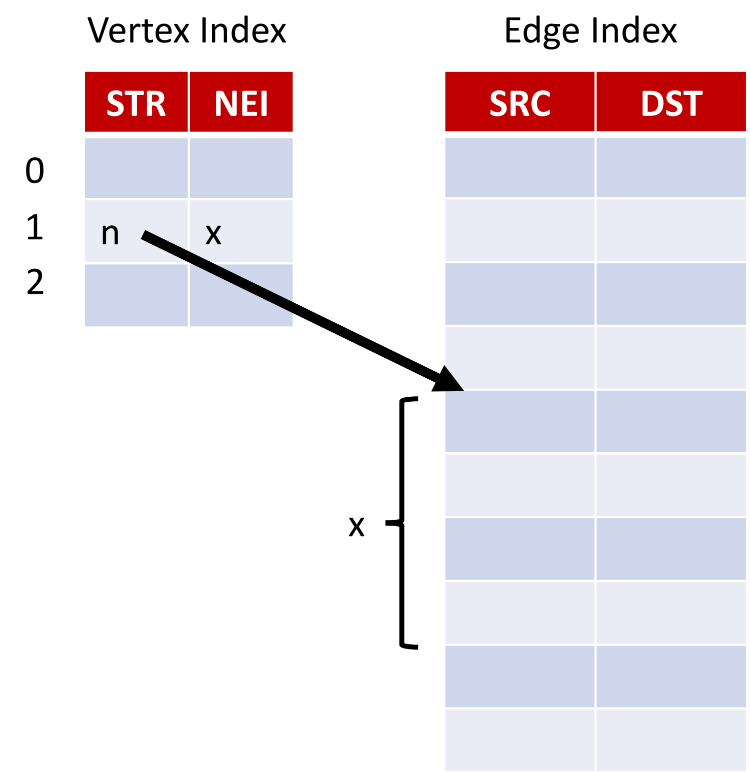


4-Truss Decomposition



How does our k-Truss work though?

Double Index Graph Structure

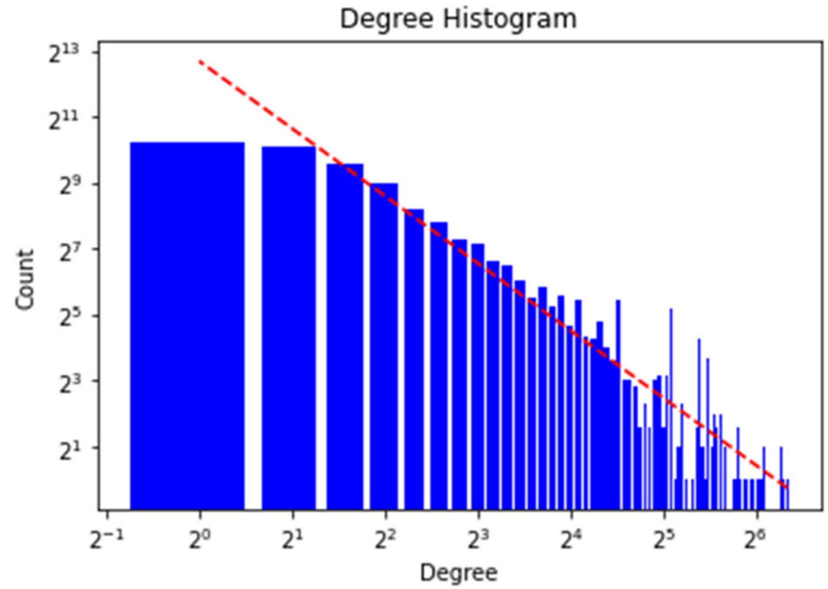


- Properties
  - O(1) time complexity
    - Locate specific vertex from given edge ID
    - Locate adjacency list from given vertex ID
  - Quickly reference adjacent edges for removal
  - Efficiently load balance by distributing across edges rather than vertices

[Bader, Du, Rodriguez 2021]

Minimized Search Kernel

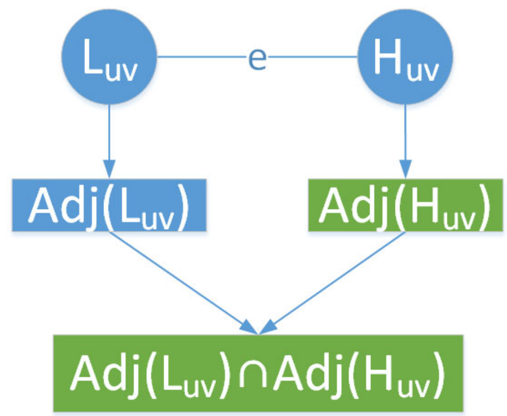
- List intersection considers the edges formed by the combination of the edgelists of vertex u, and vertex v to find a vertex w.
- By utilizing the degree of each vertex maintained in our double index graph data structure, it exploits smaller adjacency lists to reduce searches
- In skewed graphs like real world graphs which have power law distributions, this performance improvement can be distinct.
- On the right, the figure shows how most nodes have a small degree, fewer are highly connected.



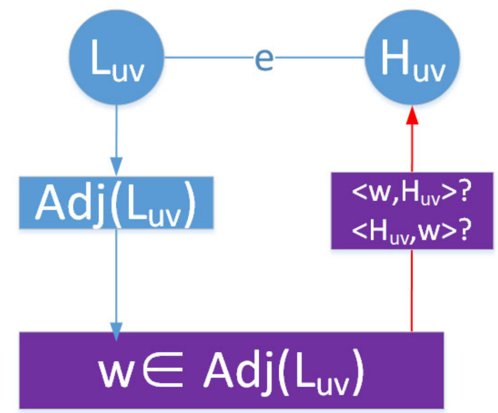
Log-log plot of Ca-GrQc distribution

Baseline Comparison (Naïve method)

- Graph is preprocessed as before, nodes with degrees too low are removed
- Utilize same parallel abstractions inherent to Chapel
- Each edge is processed and removed if it is incident to less than k-2 edges
- Once no changes have been made, the residual subgraph is returned.



List Intersection



Minimum Search

New Jersey Institute of Technology

Performance improvements

	List Intersection Naive K-Truss	MS Naive K-Truss	MS Opt K-Truss	MS Max K-Truss	MS Truss Decomposition	Speedup of LI vs. MS Naive	Speedup of LI vs. MS Opt
amazon0601	1008.58	509.29	60.61	93.22	66.22	2	16.6
as-caida20071105	16.7	2.98	1	1.83	0.88	5.6	16.7
ca-AstroPh	113.28	56.11	9.64	11.16	5.17	2	11.7
ca-CondMat	23.52	11.58	2.11	2.58	2.21	2	11.2
ca-GrQc	2.49	1.24	0.29	0.35	0.36	2	8.6
ca-HepPh	29.33	14.69	3.07	3.22	3.45	2	9.6
ca-HepTh	3.88	1.93	0.5	0.61	0.61	2	7.7
com-Youtube	4885.27	302.37	55.72	71.89	61.94	16.2	87.7
Delaunay n16	735.75	378.16	4.91	5.83	4.87	1.9	149

Conclusion

- We demonstrate that k-truss can be implemented using our double index data structure.
- We add graph structure reduction methods to the flexibility of Arkouda.
- We demonstrate that the graph data structure in Arkouda can be exploited for minimizing triangle searches in k-truss.

New Jersey Institute of Technology

Acknowledgement

We appreciate the help from Brad Chamberlain, Elliot Joseph Ronaghan, Engin Kayraklioglu, David Longnecker and the Chapel community when we integrated the algorithms into Arkouda. This research was funded in part by NSF grant number CCF-2109988.

NJIT New Jersey Institute of Technology

Thank You!

Q&A