Hierarchical Sparse Graph Computations on Multicore Platforms

Humayun Kabir Kamesh Madduri (advisor)

Computer Science and Engineering
The Pennsylvania State University

Research Motivations

Sparse graph computations are useful for social network analysis and graph analytics.

- Sparse graph computations have challenges
  - The memory access pattern of sparse algorithms is irregular
  - Degree distribution is skewed; time taken to process vertices varies
  - Sparse matrix computations have high ratio of load to operation count
- Modern computing systems are complex
  - The memory system has multiple levels
  - Increasing number of threads; threads should be kept busy to achieve performance

We need to develop sparse graph algorithms on shared memory systems that perform well.

Our Contributions

- Developed **PKC** for k-core decomposition, on average 1.90× faster than ParK
- Developed **PKT** for k-truss decomposition, average relative speedup achieved is 9.68×
- **Graph analysis** using above techniques; they are used for coarsening, sparsification, vertex reordering, partitioing and community detection.
- Developed ***CSR-k*** for sparse matrix-vector multiplication (SpMV); *CSR-k* is 3.82× and 2.12× faster than *MKL* and pOSKI respectively
- Sparse triangular solution (STS) using *CSR-k* is 2.44× and 4.21× faster than coloring and level-set respectively

Acknowledgements

I would like to thank Padma Raghavan, Joshua D. Booth, Guillaume Aupy, Anne Benoit, Yves Robert. This work is supported by the US National Science Foundation grants ACI-1253881 and CCF-1439057.

PKC: parallel k-core decomposition

- On average 2.63× faster than state-of-art ParK
- **Key optimizations:** Switches to a smaller graph - decreases scan and processing time; decreases memory requirements; decreases atomic operations

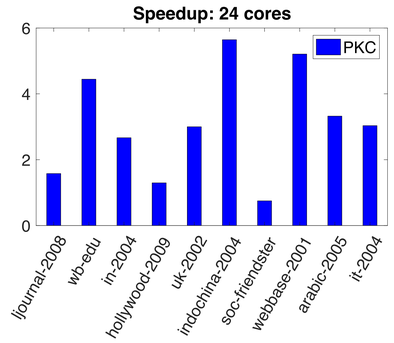

Figure 1: Speedup on 24 cores relative to ParK.

CSR-k: a multilevel data structure to store sparse matrices

- **Graph coarsening** is used to represent matrix in *CSR-k*
- **Decrease bandwidth:** RCM on the coarsened graph; induces ordering on original graph
- **SpMV** using *CSR-k* is 3.82× and 2.12× faster than **MKL** and **pOSKI**

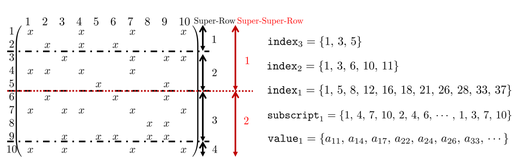

Figure 2: An example *CSR-k* representation for $k = 3$.

PKT: parallel k-truss decomposition

- An array **s** contains the support of the edges.
- Edge e_0 has trussness value 2.

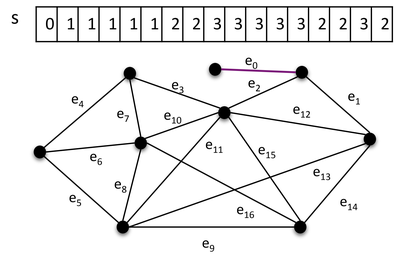

Figure 3: **Iteration 1**: Finds edges with trussness 2.

- Initially, edges e_1, e_2, e_3, e_4, e_5 have trussness 3.
- Red edges are processed; e_6, e_7 have trussness 3.

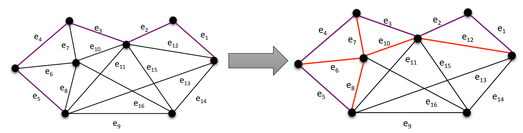

Figure 4: **Iteration 2**: Finds edges with trussness 3.

STS using *CSR-k*

- Color coarsened graph; solve color in parallel
- Increases spatial and temporal locality

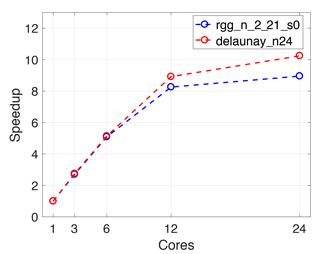

Figure 5: Relative speedup of STS-3.

Graph Analysis using PKC and PKT

- Decomposition algorithms are used for coarsening, sparsification, partitioning, community detection
- Compression friendly ordering SlashBurn: SB(k) - removes k vertices with highest centrality score
- Core (truss) based SB-core - removes all vertices in max k-core (k-truss)

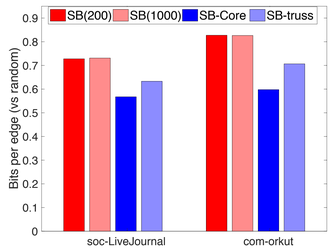

Figure 6: Bits needed per edge (vs random) to compress graph.

References

[1] H. Kabir and K. Madduri.
Parallel k-core decomposition on multicore platforms.
In *Proc. IPDPS ParSocial Workshop*, 2017.

[2] H. Kabir and K. Madduri.
Shared-memory graph truss decomposition.
In *Proc. HiPC 2017 [Short version: Graph Challenge - Student Innovation Award (HPEC 2017)]*.

[3] H. Kabir and K. Madduri.
kGD: New social network analysis methods using hierarchical graph decompositions.
(In preparation).

[4] H. Kabir, J. D. Booth, and P. Raghavan.
A multilevel compressed sparse row format for efficient sparse computations on multicore processors.
In *Proc. HiPC*, 2014.

[5] H. Kabir, J. D. Booth, G. Aupy, A. Benoit, Y. Robert, and P. Raghavan.
STS-k: a multilevel sparse triangular solution scheme for numa multicores.
In *Proc. SC15*, 2015.