Hypergraph Partitioning for Computing Matrix Powers

Erin Carson and Nick Knight

Berkeley Parlab

Research supported by Microsoft (Award #024263) and Intel (Award #024894) funding and by matching funding by U.C. Discovery (Award #DIG07-10227). Additional support comes from Par Lab affiliates National Instruments, NEC, Nokia, NVIDIA, Samsung, and Sun Microsystems.

Motivation

- Communication is the performance bottleneck in many algorithms
  - Gap in communication/computation costs growing exponentially
- Krylov Subspace Methods are commonly used for solving linear systems
  - Standard implementations are communication-bound due to required SpMV and orthogonalization in every iteration
  - Solution: rearrange algoroithms to perform s iterations at a time without communicating (s-step methods)
  - SpMV in each iteration is replaced with a call to the Matrix Powers Kernel, which performs s SpMVs while reading the matrix only once
    - Used to generate s basis vectors for the Krylov Subspace

$$\mathcal{K}_k(A, v) = span\{v, Av, ..., A^{k-1}v\}$$

The Matrix Powers Kernel

- Assuming the matrix is well-partitioned, we can perform s repeated SpMVs (linear relaxation) reading A only once
- How can we find a good partition of A for this computation?
- Current approach
  - Graph partition A+A_T
  - Problems:
    - Not sufficient if A is highly unsymmetric
    - Graph partitioning does not accurately count communications (citation)
    - Only captures dependencies for 1 SpMV

Hypergraph Formulation

- Solution: encode matrix powers dependencies in a hypergraph
- For SpMV: column-net model – hyperedge for each column, vertex for each row. Vertex is in hyperedge if there is a nonzero in that row, col
- New idea: k-level Column Nets
  - To represent s SpMVs, union column nets via BFS: s steps of the transitive closure
- If we construct a hypergraph using our k-level Column Net model, the cost of a p-way partition is exactly the communication required between p processors for the matrix powers computation using that partition

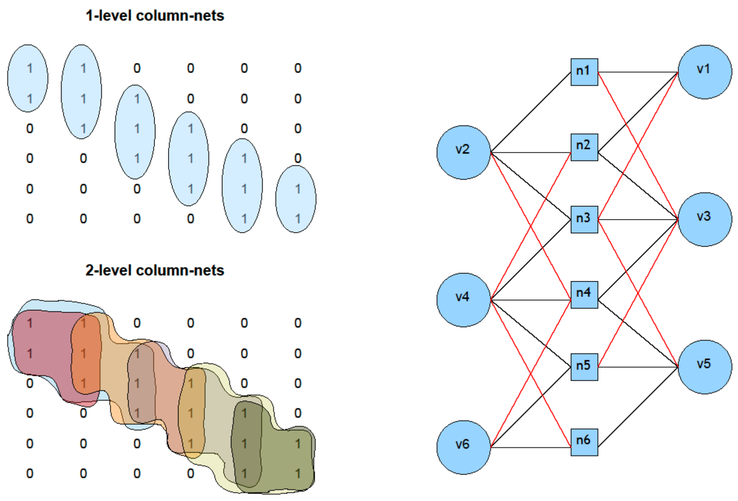
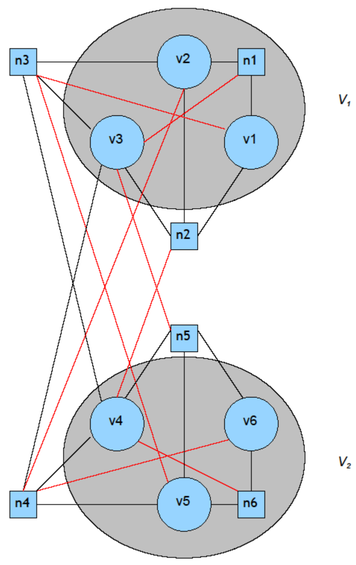


Figure: Example of k-level column net model for a tridiagonal matrix. On the right, we see the equivalent hypergraph. Black lines represent dependencies for Ax, red lines represent additional dependencies for computing (A^2)x.

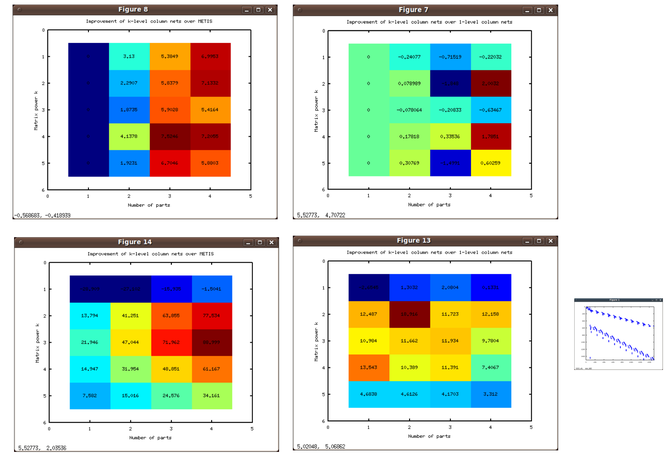
Partitioning



Figures: Depiction of a 2-way partition of the hypergraph above. In this example, nets 1 and 6 have connectivity 1 (internal), whereas the other nets have connectivities of 2 (external).

- Hypergraph partitioning is performed using the PaToH software(http://bmi.osu.edu/~umit/software.html)
- "External nets" are those which have vertices in more than one partition
- It can be shown that the problem of finding a partition which minimizes communication in a matrix powers computation is polynomial-time reducible to finding the minimum p-way cut of the constructed hypergraph.
  - In other words, the connectivities of the nets count the data that must be communicated across partitions.

Results



Figures: Percent Reduction in Total Communication Volume over Metis (left) and over 1-level nets (right). Shown for 5-pt stencil (top) and unsymmetric matrix (bottom).

Using Heuristics

- Constructing the full k-level column nets is a significant preprocessing cost
- Can we reduce the cost of constructing and partitioning the hypergraph by using heuristics?
- Some ideas:
  - Sparsification of the input matrix
    - Dropping rows with many nonzeros during construction or dropping large nets from consideration when partitioning
    - Approximating k-level column nets with 1-level column nets + improvement by iterative swapping

Future Work

- Implementation and improvement of heuristic strategies
- Incorporate load balancing costs into hypergraph formulation
  - Involves solving a multi-constraint hypergraph partitioning problem
- Experiment with 2D partitions (involves communication in between levels of the matrix powers kernel)
  - Provides greater flexibility in reducing overall communication volume