SIAM CSE'09 Minisymposium on
Parallel Sparse Matrix Computations and Enabling Algorithms
March 2, 2009, Miami, FL

Partitioning Sparse Matrices

Ümit V. Çatalyürek

Associate Professor

Department of Biomedical Informatics

Department of Electrical & Computer Engineering

The Ohio State University

Joint work with

Cevdet Aykanat (Bilkent University, Ankara, Turkey)

Bora Uçar (LIP-ENS Lyon)

Matrix Partitioning

- Enabling inherent parallelization in solvers
  - mathematical programming
  - LU factorization
  - QR factorization
- Improving performance of sparse matrix-vector multiply
- Finding fill-reducing orderings via nested-dissection

- Also, Sparse Matrix is a good abstraction for modeling input-output interactions, can be used to model the workload and data partitioning of many other applications!

Hypergraphs: Matrix Partitioning models/methods

- Matrix partitioning:
  - 1D partitioning (by rows or by columns),
  - 2D partitioning (jagged, checkerboard, nonzero-based, or orthogonal recursive).
- Hypergraph models (vertices represent data units/computations to partition, hyperedges represent dependencies): row-net, column-net, fine-grain (nonzero-based) models.
- Partitioning algorithms based on hypergraph models: Rowwise, columnwise, nonzero-based, jagged-like, checkerboard-like, Mondrian.
- Key point: two useful cutsize definitions matches
  - the total communication volume in $y \leftarrow Ax$ computations,
  - minimization of border size (number of rows/columns in the border).

Hypergraph

- A hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{N})$ is a set of vertices $\mathcal{V}$ and a set of hyperedges (nets) $\mathcal{N}$.
- A net $n \in \mathcal{N}$ is a subset of vertices.
- A cost $c(n)$ is associated with each net n.
- A weight $w(v)$ is associated with each vertex v.
- An undirected graph can be seen as a hypergraph where each net contains exactly two vertices.

Hypergraph Partitioning

- K-way hypergraph partition: $\Pi=\{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K\}$
  - $\mathcal{V}_k$ is nonempty subset of $\mathcal{V}$, i.e., $\mathcal{V}_k \subseteq \mathcal{V}$,
  - parts are pairwise disjoint, i.e., $\mathcal{V}_k \cap \mathcal{V}_l = \emptyset$,
  - union of K parts is equal to $\mathcal{V}$, i.e., $\bigcup \mathcal{V}_k = \mathcal{V}$.
- In Π
  - a net that has at least one pin in a part is said to *connect* that part
  - *connectivity* $\lambda(n)$ of a net n is the number of parts connected by n
- Objective:
  - minimize cutsize(Π) = $\sum_{n \in N} c(n)\,(\lambda(n) - 1)$ or
  - minimize cutsize(Π) = $\sum_{n \in N \wedge \lambda(n) > 1} c(n)$
- Constraint:
  - $\mathcal{W}_k \leq \mathcal{W}_{avg}(1 + \varepsilon)$ where $\mathcal{W}_k$: weight of part $\mathcal{V}_k$, ε : max. imbalance ratio

Hypergraph Partitioning

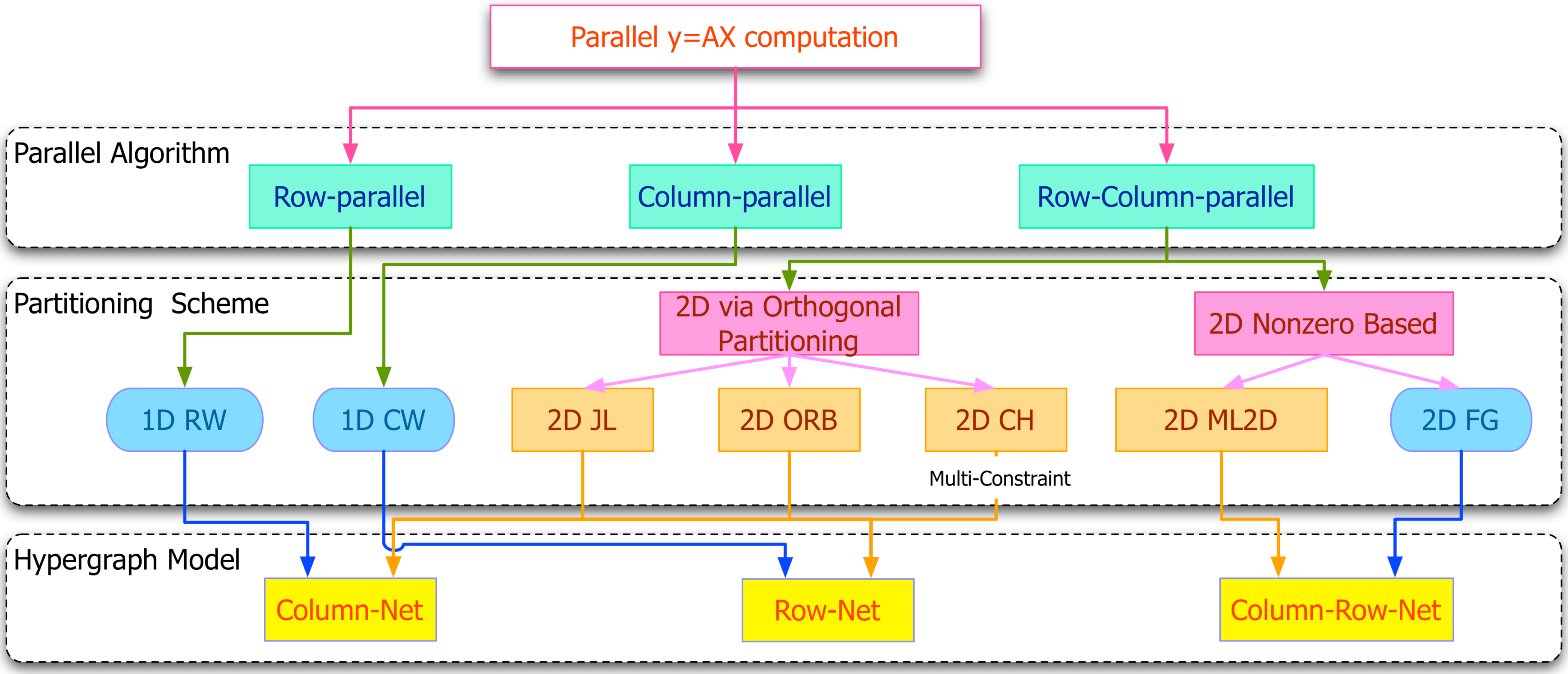
Tools

- hMETIS (Karypis and Kumar, Univ. Minnesota),
- MLPart (Caldwell, Kahng, and Markov, UCLA/UMich),
- Mondriaan (Bisseling and Meesen, Utrecht Univ.),
- Parkway (Trifunovic and Knottenbelt, Imperial Coll. London),
- PaToH (Catalyurek and Aykanat, Bilkent Univ.),
- Zoltan-PHG (Devine, Boman, Heaphy, Bisseling, and Catalyurek, Sandia National Labs.).

Applications

- VLSI: circuit partitioning,
- Scientific computing: matrix partitioning, ordering, cryptology, etc.,
- Parallel/distributed computing: volume rendering, data aggregation, declustering/clustering, scheduling,
- Software engineering, information retrieval, processing spatial join queries, etc.

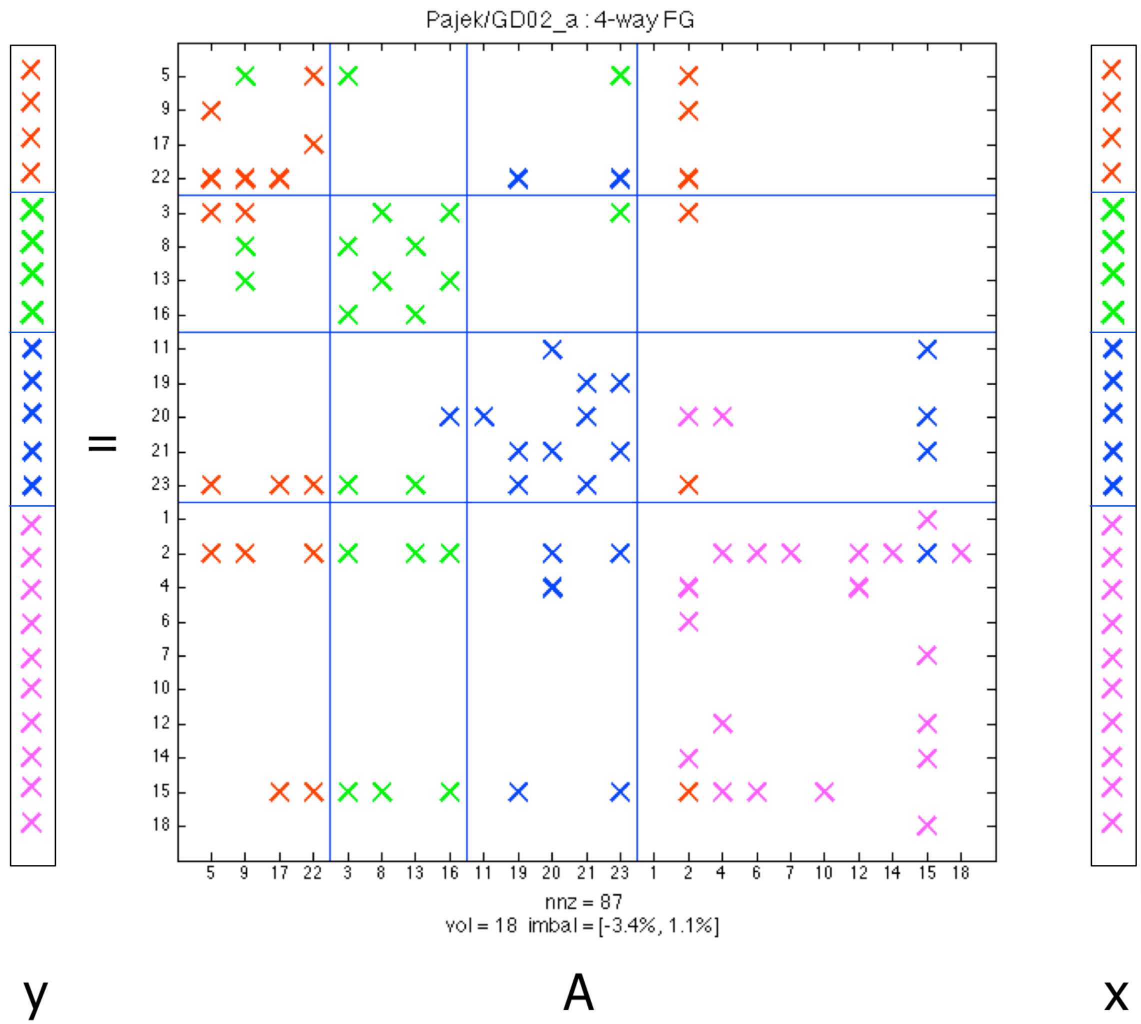
Taxonomy of Sparse Partitioning Models and Methods



Row-Column Parallel Matrix-Vector Multiply

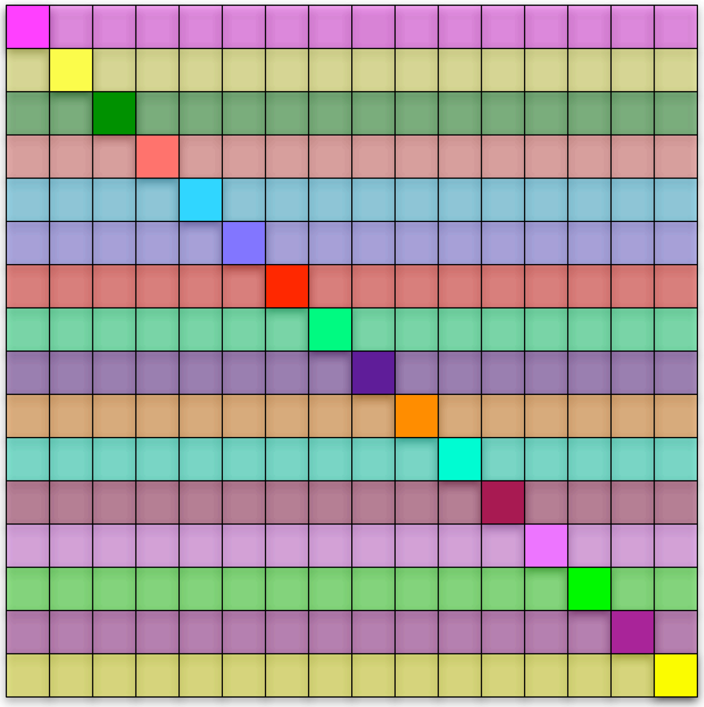
1. Send the local input-vector entries x_j to those processors that has at least one nonzero in column j.
2. Compute the scalar products $a_{ij}\ x_j$ for the local nonzeros, i.e., the nonzeros for which $map(a_{ij}) = P_k$ and accumulate the results y^k_i for the same row index i.
3. Send local nonzero partial results y^k_i to the processor $map(y_i) \neq P_k$
4. Add the partial y^ℓ_i results received to compute the final result $y_i = \Sigma\ y^\ell_i$ for which $map(y_i) = P_k$.

Row-Columns Parallel Matrix-Vetor Multiply

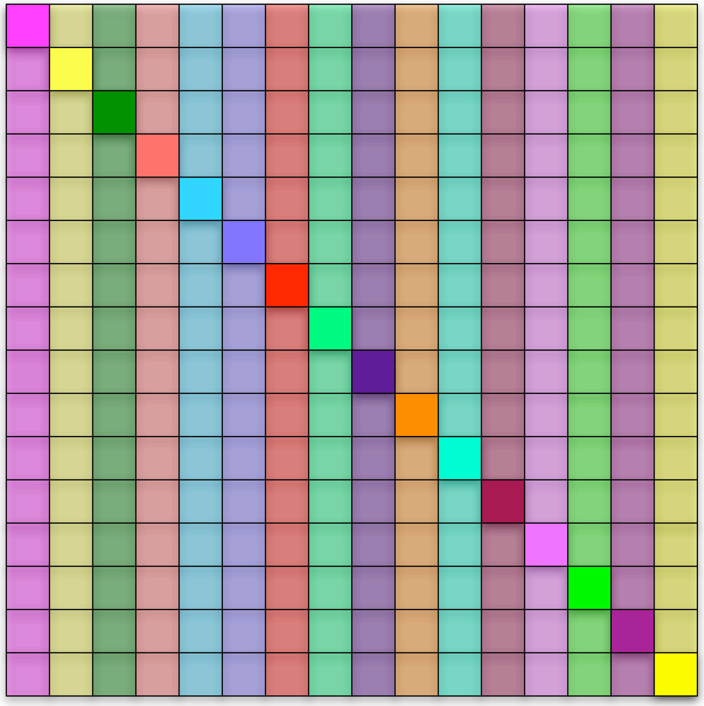


1D Partitioning

1D Row-wise Partitioning



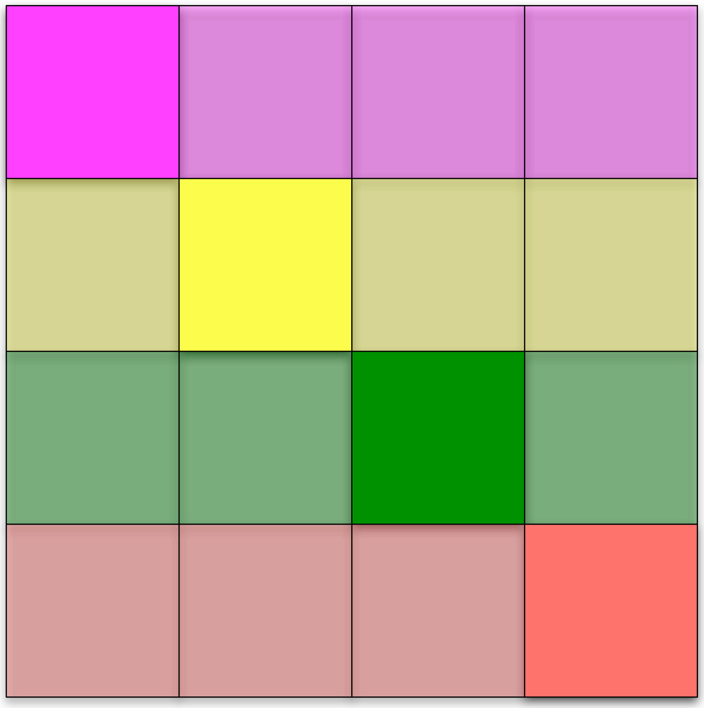
1D Column-wise Partitioning



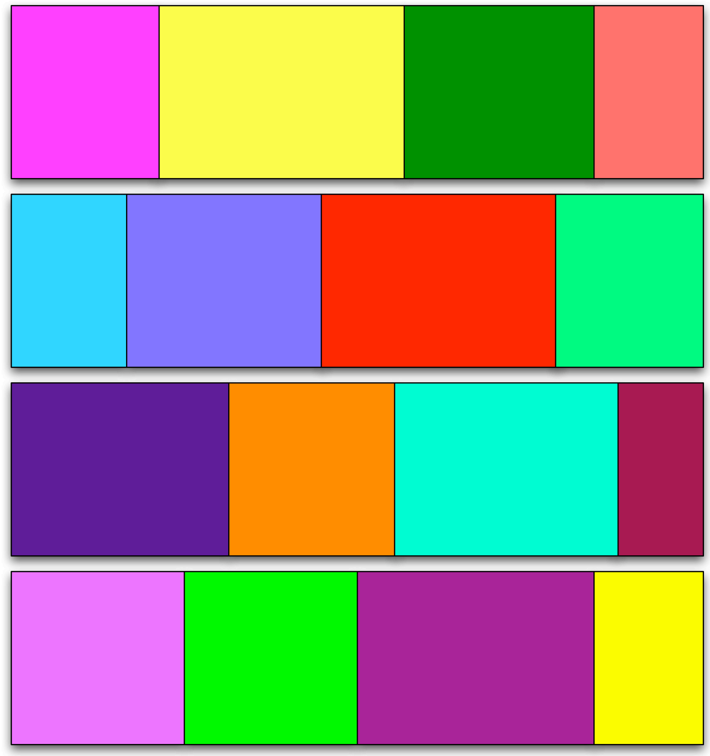
- M x N matrices with K processors
- Worst case
  - Total Volume = (K-1) x N words or (K-1) x M words
  - Total Number Messages = K x (K-1)

2D Partitioning: Jagged-Like

2D Jagged-Like Partitioning

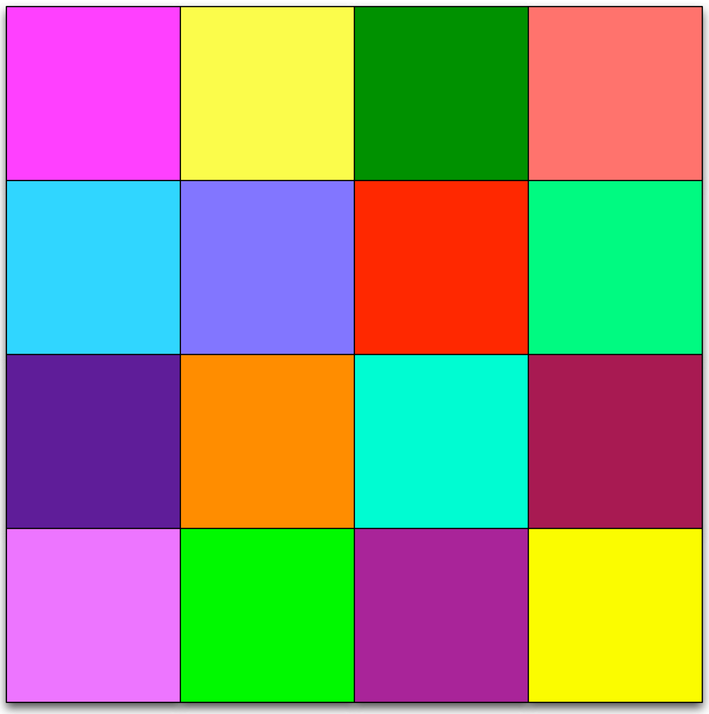


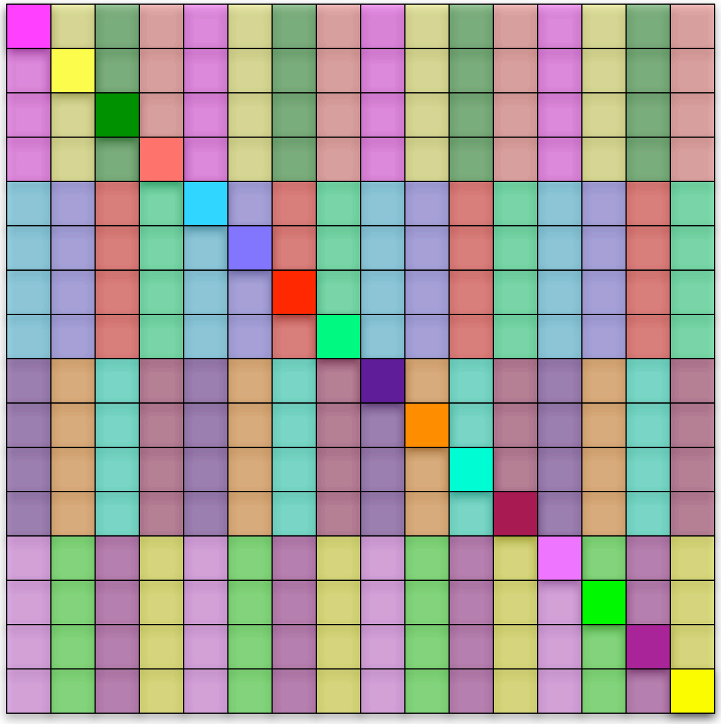
2D Jagged-Like Partitioning



- M x N matrices with K=PxQ processors
- Worst case
  - Total Volume = (K-P) x N + (Q-1) x M
  - Total Number Messages = K x (K-Q) + K x (Q-1) = K x (K-1)

2D Partitioning: Checkerboard





- M x N matrices with K=PxQ processors
- Worst case
  - Total Volume = (P-1) x N + (Q-1) x M
  - Total Number Messages = P+Q-2

PaToH: *Par*titioning *To*ols for *H*ypergraphs

- PaToH provides a set of algorithms that implement state-of-the-art multilevel, recursive bisection based hypergraph partitioning [Catalyurek and Aykanat, Tech.Rep(1999)].
- Features include:
  - simple hypergraph partitioning,
  - minimize the cutsize based on connectivity (the formula before) or just the weighted sum of the cost of the cut nets (e.g., the number of split columns and/or rows in a matrix),
  - fixed-vertex regime (some vertices are fixed to certain parts),
  - multi-constraint partitioning (vertices have a set of weights).

PaToH: MATLAB interface

Hypergraph partitioning interface:

```
[partvec [, ptime]] = PaToH(H, K [,nconst, cwghts, nwghts])
% H: hypergraph in a matrix form (columns are hyperedges)
% K: the number of parts
% nconst: number of constraints (for multi-constraint partitioning)
% cwghts, nwghts: vertex weights and hyperedge costs
```

PaToH: MATLAB interface

We provide matrix partitioning interface:

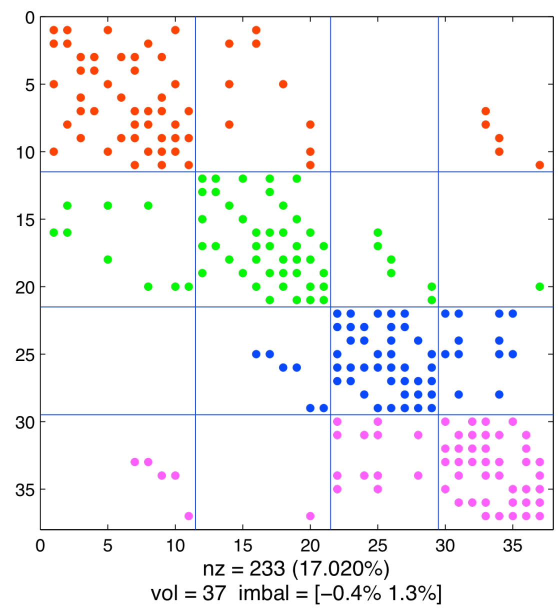
```
[outpv, inpv, nnzpv, ptime] = PaToHMatrixPart(A, K, dim)
%outpv, inpv, nnzpv: part vectors for y, x, and A of y=Ax
%A: a matrix %K: the number of parts
%dim: a string (RW(U|S), CW, FG, JL, CL)
```

A function to display the partitionings (inspired by the spypart of J. Gilbert and S.-H. Teng)

```
PaToHSpy(nnzpv [, K, outpv, inpv])
%K: part number
%outpv, inpv, nnzpv: as returned by PaToHMatrixPart
```

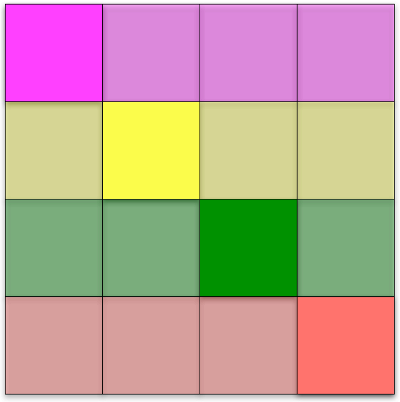
PaToH: MATLAB interface examples

```
>> p= UFget('vanHeukelum/cage5'); %from UFL
>> [outpv, inpv,nnzpv,ptime] = PaToHMatrixPart(p.A, 4, 'RWS');
>> PaToHComputeVolume(outpv, inpv, nnzpv, 4)
ans=
    37
>> PaToHSpy(nnzpv, 4, outpv, inpv);
```

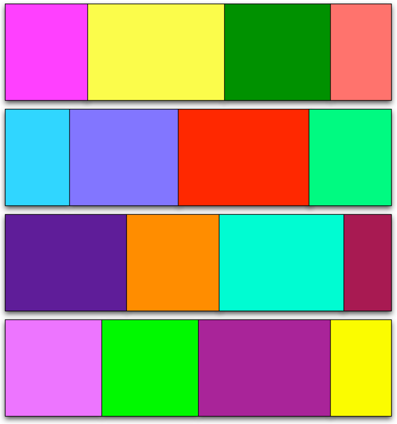


PaToH: MATLAB interface examples

```
function[outpv,inpv,nnzpv,ptime]=jaggedSymPart(A,k1,k2)
```




```
prow= rowwiseSymPart(A, k1);%Phase 1
```

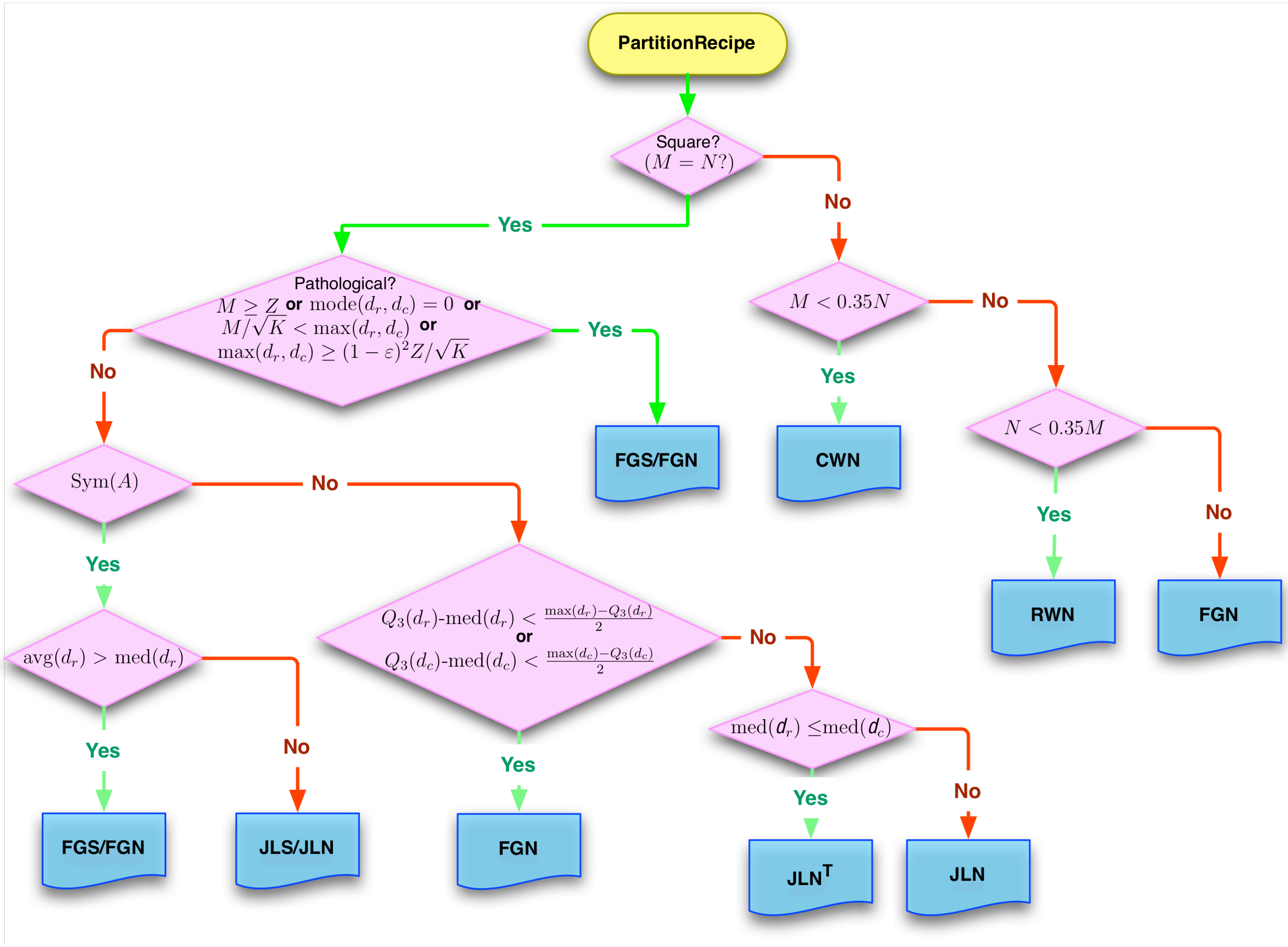



```
for submat = 1:k1, %Phase 2 for a block
  rowIndices = find(prow == submat);
  Asub= A(rowIndices,1:n);
  [ign1,ign2,partsub,timesub] =
              colwiseUnsymPart(Asub, k2);
  ...
end
```

Experimental Results

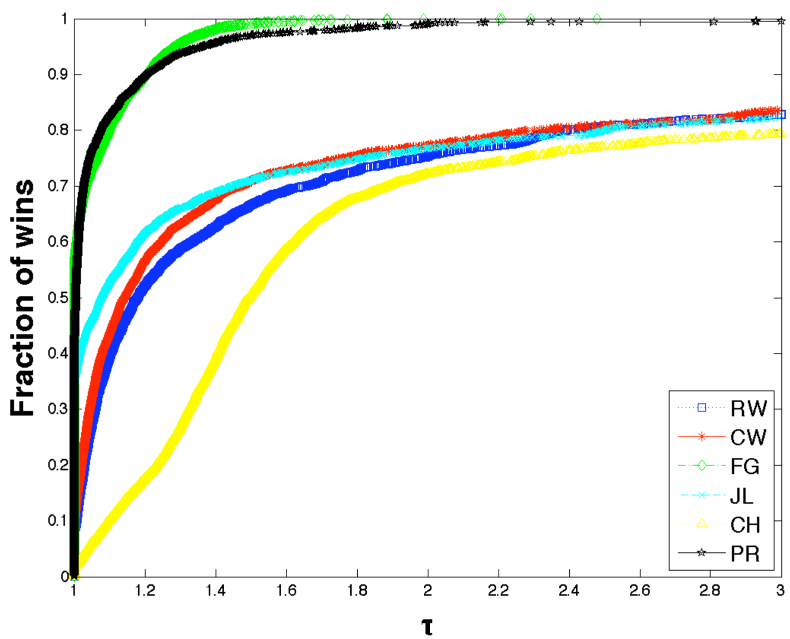
- Tested 1,413 matrices (out of 1,877) from UFL Collection
  - #rows >= 500 and #columns >= 500
  - #non-zeros < 10,000,000
- K-way partitioning for K = 4, 16, 64 and 256
  - If 50 x K >= max {#rows, #columns}
- Partitioning instance = matrix & K
  - For each partitioning instance we run RW, CW, JL, CH, FG methods
- The method **PR** chooses among RW, CW, FG, and JL according to some matrix characteristics
- Sequential runs on a Linux Cluster
  - 64 dual 2.4GHz Opteron CPUs, 8GB ram

A Recipe for Matrix Partitioning

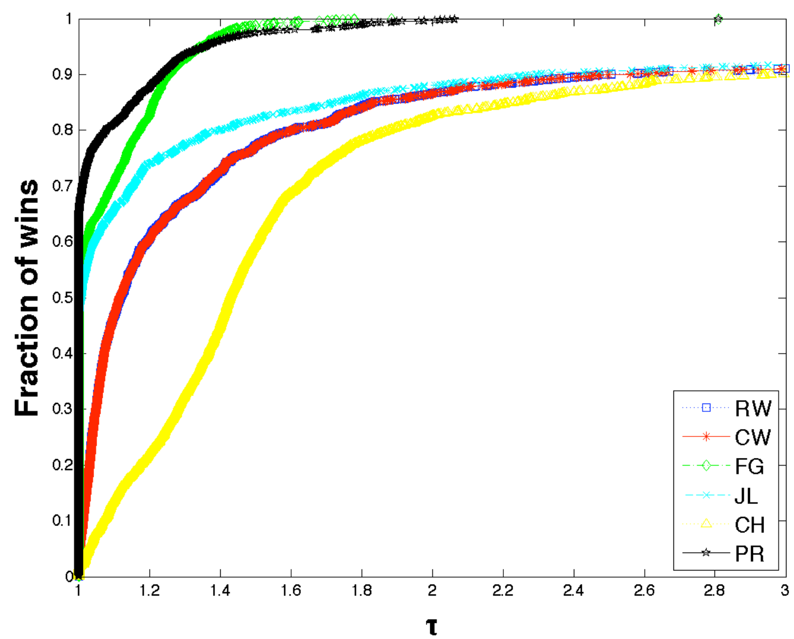


Experimental Results: Total Communication Volume

Performance Profiles

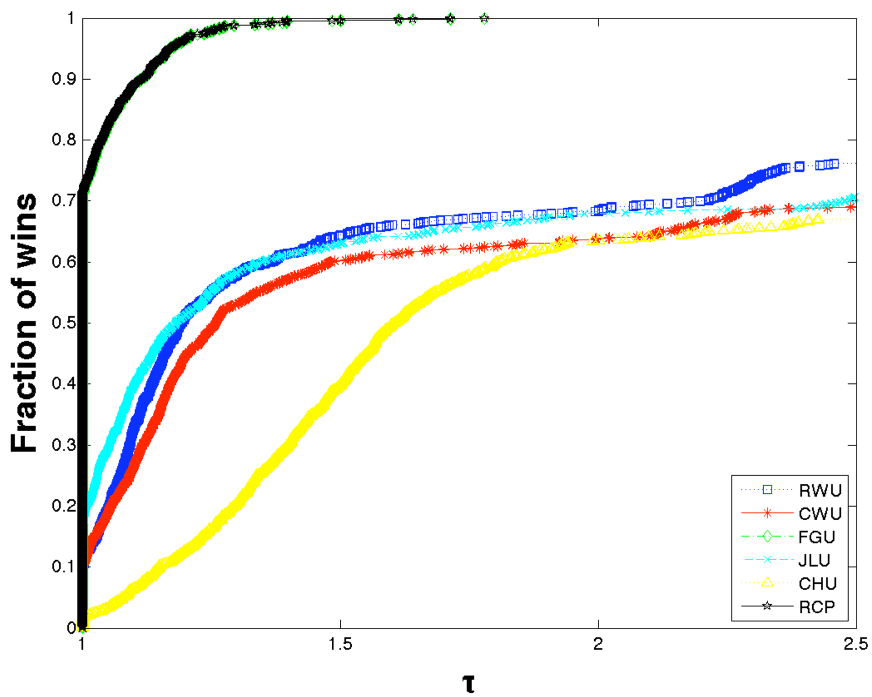


All Instances (4100)

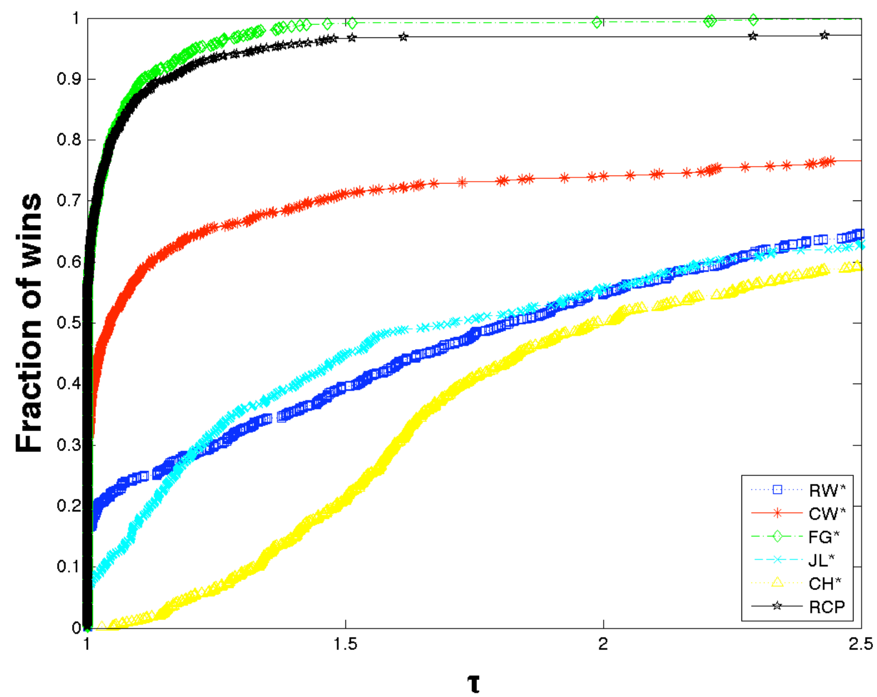


Square Symmetric (1932)

Experimental Results: Total Communication Volume

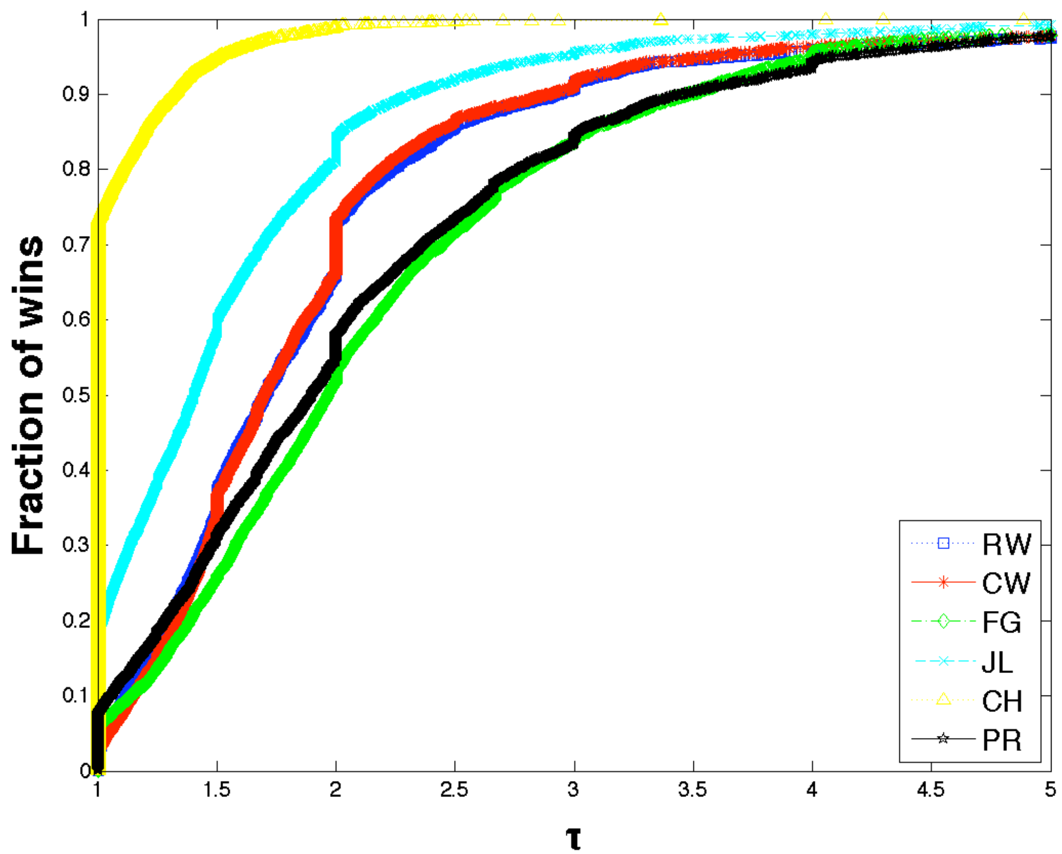


Square Non-symmetric (1456)



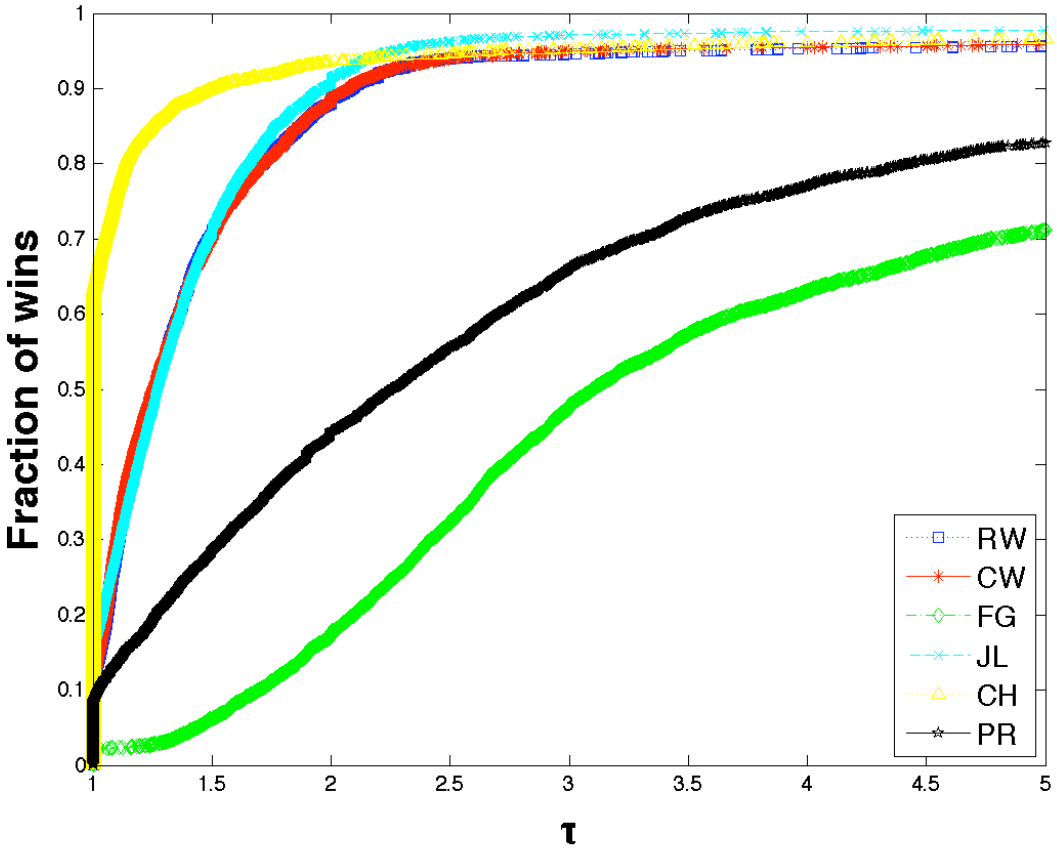
Rectangular (712)
N>M (667)
M>N (45)

Experimental Results: Total Number of Messages



- Checkerboard and jagged approaches are preferable to others.
- The fine-grain approach: always a higher number of messages.
- PR is very close to fine-grain: most of the time chooses fine-grain approach.

Experimental Results: Execution Time



- fine-grain is the slowest.
- checkerboard is the fastest.
- PR is faster than the fine-grain with similar performance (previous slide).

Summary and Future Plans

- Hypergraph models for Matrix Partitioning
  - Well.. some are not new but not have been adopted by applications yet. Why? (Information dissemination problem? Tool?)
- Developed a matrix partitioning interface to PaToH in MATLAB.
  - Provides rapid prototyping of new partitioning algorithms
  - With UFget, enabled extensive experiments on
  - Will be available soon!
- Results:
  - FG almost always yields smaller total volume of communication
  - In rectangular cases, 1D partitioning along the longer dimension is an acceptable alternative (concurs with R. Bisseling's findings).
  - For square symmetric matrices, jagged partitioning approach is sometimes the best (all metrics included).
  - PR is faster than FG with similar performance
- Work in progress
  - Parallel Matrix Partitioning via Zoltan

Thanks

- Contact Info:
  - umit@bmi.osu.edu
  - http://bmi.osu.edu/~umit
- Also:
  - http://www.cs.sandia.gov/Zoltan/
  - http://www.cscapes.org/
- Acknowledgements:
  - The work was supported in part by the National Science Foundation under Grants CNS-0643969, CCF-0342615, CNS-0403342 and DoE SciDAC Grant DE-FC02-06ER2775, Agence Nationale de la Recherche through SOLSTICE project ANR-06-CIS6-010, and (TUBITAK) under project EEEAG-106E069.