Using hypergraph partitioning for iterative linear system solution techniques

Masha Sosonkina* Yousef Saad† Bora Uçar‡

June 13, 2006

1 Method description

Bipartite graph [3] and hypergraph [1] based models and partitioning techniques have become common tools to partition general sparse matrices for parallel processing. These two models can represent non-symmetric matrices and can produce non-symmetric partitions. Traditionally, the corresponding partitioning techniques target minimizing communication overhead while maintaining load balance during parallel execution of communication-intensive (fine grain) computations such as matrix-vector multiplies.

In this talk, we focus on the effect of the hypergraph partitioning on the convergence of iterative methods. This task requires considering not only the sparsity pattern of a matrix, but also its numerical properties. In particular, one has to determine which numerical properties to choose and how to incorporate them into the models to be handled by the partitioning algorithms. A common framework to incorporate extra properties into the hypergraph models is to associate weights with the hyperedges and/or vertices. Our first approach is to use a weighted hypergraph $G_{Hw} = (V, H, W_v, W_h)$, where $V = \{v_1, \ldots, v_n\}$ is the set of vertices representing rows of an $n \times n$ matrix A; $H = \{h_1, \ldots, h_n\}$ is the set of hyperedges representing the columns, such that h_j contains the vertices $\{v_i \mid a_{ij} \in A \text{ and } j = 1, \ldots, n\}$ [1]; W_v is the set of vertex weights (here we use unit weights $W_v(i) = 1$ for $i = 1, \ldots, n$); W_h is the set of hyperedge weights. The weight $W_h(j)$ of the hyperedge h_j is based on the notion of the *weak diagonal dominance* of the matrix columns. The hyperedge h_j is deemed as type $\boldsymbol{\mathcal{D}}$ if the corresponding matrix column is weakly diagonal dominant, i.e., $|a_{jj}| \geq \sum_{i \neq j} |a_{ij}|$, and as type non$\boldsymbol{\mathcal{D}}$ otherwise. We explore different weighting schemes in which either $\boldsymbol{\mathcal{D}}$ or non$\boldsymbol{\mathcal{D}}$ hyperedges are heavily weighted. This approach tunes the partitioning algorithms to favor $\boldsymbol{\mathcal{D}}$ or non$\boldsymbol{\mathcal{D}}$ hyperedges. We consider several possibilities for weight values.

Our second approach in incorporating the numerical properties is to separate the original (unweighted) hypergraph into two subhypergraphs—one with the $\boldsymbol{\mathcal{D}}$ hyperedges (columns) and the corresponding vertices (rows) and the other with the non$\boldsymbol{\mathcal{D}}$ hyperedges

*Ames Laboratory/DOE, Iowa State University, Ames, IA 50011, (masha@scl.ameslab.gov).

†Department of Computer Science and Engineering, University of Minnesota, 200 Union Street S.E., Minneapolis, MN 55455, (saad@cs.umn.edu).

‡Department of Mathematics and Computer Science, Emory University, Atlanta, GA 30322 (ubora@mathcs.emory.edu).

Table 1: Matrix characteristics

Matrix	n	nnz	condest	% $\mathcal{D}$ row; $\mathcal{D}$ col
igbt3	10,938	234,006	4.74e19	26 ; 1
bjtcai	27,628	442,898	6.46e19	43 ; 49
mat9	103,430	2,121,550	3.08e23	35 ; 38
new3	125,329	2,678,750	3.48e22	55 ; 63

Table 2: Iteration numbers for various 8-way partitioning options

Matrix	nW	$\mathcal{D}$-heavy	non$\mathcal{D}$-heavy	SPM
igbt3	50	37	41	34
bjtcai	79	78	132	52
mat9	81	66	82	97
new3	98	94	96	66

(columns) and the corresponding vertices (rows). Then, partition *separately* the $\mathcal{D}$ and non$\mathcal{D}$ subhypergraphs into the equal number p of partitions, where p is the number of the final partitions. In this case, a bipartite graph consisting of $2p$ vertices is obtained and may be input to a matching algorithm which will match a $\mathcal{D}$ part to a non$\mathcal{D}$ one based on the degree of connectivity in the bipartite graph.

2 Numerical experiments

The test matrices are shown in Table 1. They originate from circuit device simulation and are provided in the University of Florida sparse matrix collection by O. Schenk. We used PaToH [2] to partition the matrices and a few methods form pARMS library [4] to solve the distributed linear systems. Table 2 presents some preliminary results. In particular, it shows the number of iterations achieved with different applications of PaToH: the column nW corresponds to the traditional hypergraph model, the column $\mathcal{D}$-heavy corresponds to the model in which $\mathcal{D}$ hyperedges have heavy weights, the column non$\mathcal{D}$-heavy corresponds to the model in which non$\mathcal{D}$ hyperedges have heavy weights, and the column SPM corresponds to separate partitioning followed by matching. The linear systems were partitioned to 8 processors and solved using flexible GMRES(20) preconditioned with non-overlapping Additive Schwarz to reduce the residual norm by 10^{-5}.

References

[1] Ü. V. Çatalyürek and C. Aykanat. Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication. *IEEE Trans. Parallel Distrib. Syst.*, 10(7):673–693, 1999.

[2] Ü. V. Çatalyürek and C. Aykanat. PaToH: A multilevel hypergraph partitioning tool, version 3.0. Technical report, Bilkent University, Department of Computer Engineering, Ankara, 06533 Turkey, 1999.

[3] B. Hendrickson and T. G. Kolda. Graph partitioning models for parallel computing. *Parallel Computing*, 26(12):1519–1534, 2000.

[4] Z. Li, Y. Saad, and M. Sosonkina. pARMS: A parallel version of the algebraic recursive multilevel solver. *Numerical Linear Algebra with Applications*, 10:485–509, 2003.