A Hypergraph-Partitioning Approach for Coarse-Grain Decomposition*

Ümit V. Çatalyürek
Department of Biomedical Informatics
The Ohio State University
Columbus, OH 43210
umit@cs.umd.edu

Cevdet Aykanat
Computer Engineering Department
Bilkent University
Ankara, 06533 Turkey
aykanat@cs.bilkent.edu.tr

ABSTRACT

We propose a new two-phase method for the coarse-grain decomposition of irregular computational domains. This work focuses on the 2D partitioning of sparse matrices for parallel matrix-vector multiplication. However, the proposed model can also be used to decompose computational domains of other parallel reduction problems. This work also introduces the use of multi-constraint hypergraph partitioning, for solving the decomposition problem. The proposed method explicitly models the minimization of communication volume while enforcing the upper bound of $p + q - 2$ on the maximum number of messages handled by a single processor, for a parallel system with $P = p \times q$ processors. Experimental results on a wide range of realistic sparse matrices confirm the validity of the proposed methods, by achieving up to 25 percent better partitions than the standard graph model, in terms of total communication volume, and 59 percent better partitions in terms of number of messages, on the overall average.

1. INTRODUCTION

The standard graph partitioning approach has been widely used to decompose irregular domains for the sake of efficient parallel execution [2, 3, 16, 17, 19, 22, 23, 25, 30, 34]. In our previous works [6, 7, 9], we introduced two computational hypergraph models for the decomposition problems. In the mentioned works, the decomposition problem is modeled as a mincut graph/hypergraph partitioning problem, such that minimization of cut-cost targeted to the minimization of total communication volume. However, for architectures with high message latency, minization of number of messages is also important. In this work, we propose a two-phase method which employs multi-constraint hypergraph partitioning. The proposed method naturally maintains an upper bound for the maximum number of messages handled by a single processor while minimizing the total communication volume. This upper bound is $p + q - 2$ messages per processor for a parallel system with $P = p \times q$ processors.

* This work is partially supported by Turkish Science and Research Council under grant EEEAG-199E013.

SC2001 Nov 2001, Denver (c) 2001 ACM 1-58113-293-X/01/0011 $5.00.

Partitioning of irregularly sparse matrices for repeated sparse-matrix vector multiplication (SpMxV) is an important application for the decomposition methods. Repeated SpMxV $y = Ax$ that involves large, sparse, structurally symmetric or nonsymmetric square matrix A is the kernel operation in iterative solvers. These algorithms also involve linear operations on dense vectors. For efficient parallelization of these iterative algorithms, nonzeros of matrix A should be partitioned among processors in such a way that communication overhead is kept low while maintaining computational load balance. In order to avoid the communication of vector components during linear vector operations, a symmetric partitioning scheme is adopted. That is, all vectors (including x and y vectors) used in the solver are divided conformally.

Parallel SpMxV is one of the basic parallel reduction algorithms. Elements of x vector are the inputs of the reduction and elements of y vector are the outputs of the reduction. Matrix A corresponds to the mapping matrix from input elements to output elements. Hence, any technique used in sparse matrix partitioning is also applicable to other reduction problems. For example, analysis of large scientific datasets involves reduction operations. Vast amounts of datasets are being continuously produced by advanced sensors attached to instruments; such as earth-orbiting satellites [31, 32, 37] and medical instruments [1]. In addition, long running scientific simulations that periodically generate snapshots of their time dependent states [11, 29, 36] are generating unprecedently large volumes of data for scientists. Typical analysis of such datasets usually involves subseting (to extract the data of interest from all the data available), and processing and transforming it into a new data product. The type of data processing usually results in a significant reduction in data volume. Both the existing graph models and our hypergraph models can be used to model those reduction operations as it has been used in Active Data Repository (ADR) [12, 24]. ADR is a framework developed at University of Maryland that provides support for integrating storage, retrieval and processing of multi-dimensional datasets on distributed memory machines. A variant of our fine-grain hypergraph model [5, 9] has been used in the query scheduling phase of ADR to solve decomposition (workload distribution) problem [13]. Hence the proposed two-phase coarse-grain decomposition method is also applicable to the reduction problems together with its nice number of message feature.

The application of the proposed method to the sparse-matrix partitioning problem results in 2D checkerboard partitioning. The nice property of this partitioning scheme is that it bounds the the maximum number of messages handled by a processor to $p + q - 2$. The parallel SpMxV algorithms proposed by Hendrickson et al. [18] and Lewis and van de Geijn [28] for 2D checkerboard partitioning are typically suitable for dense matrices or sparse matrices with struc-

tured nonzero patterns that are difficult to exploit. The 2D checkerboard partitioning schemes proposed in the literature [18, 27, 28, 33] do not exploit sparsity for reducing communication volume, and hence, the total communication volume may be as high as $(p+q-2)m$ for an $m \times m$ matrix. In this work, we propose a two-phase checkerboard partitioning method based on multi-constraint hypergraph partitioning for minimizing communication volume while maintaining computational load balance.

2. PRELIMINARIES

A hypergraph $\mathcal{H}=(\mathcal{V},\mathcal{N})$ is defined as a set of vertices $\mathcal{V}$ and a set of nets (hyperedges) $\mathcal{N}$ among those vertices. Every net $n_j \in \mathcal{N}$ is a subset of vertices, i.e., $n_j \subseteq \mathcal{V}$. The vertices in a net n_j are called its *pins*. Weights can be associated with the vertices of a hypergraph. Let w_i denote the weight of vertex $v_i \in \mathcal{V}$.

A *K-way vertex partition* $\Pi = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K\}$ of $\mathcal{H}$ is said to be balanced if each part $\mathcal{V}_k$ satisfies the *balance criterion*

$$W_k \leq W_{avg}(1+\varepsilon), \quad \text{for } k = 1, 2, \ldots, K. \tag{1}$$

In (1), weight W_k of a part $\mathcal{V}_k$ is defined as the sum of the weights of the vertices in that part (i.e. $W_k = \sum_{v_i \in \mathcal{V}_k} w_i$), $W_{avg} = (\sum_{v_i \in \mathcal{V}} w_i)/K$ denotes the weight of each part under the perfect load balance condition, and ε represents the predetermined maximum imbalance ratio allowed.

In a partition Π of $\mathcal{H}$, a net that has at least one pin (vertex) in a part is said to *connect* that part. *Connectivity set* Λ_j of a net n_j is defined as the set of parts connected by n_j. *Connectivity* $\lambda_j = |\Lambda_j|$ of a net n_j denotes the number of parts connected by n_j. A net n_j is said to be *cut* (*external*) if it connects more than one part (i.e. $\lambda_j > 1$), and *uncut* (*internal*) otherwise (i.e. $\lambda_j = 1$). The set of external nets of a partition Π is denoted as $\mathcal{N}_E$. There are various *cutsize* definitions for representing the cost of a partition Π. Two relevant definitions are:

$$cutsize(\Pi) = |\mathcal{N}_E| \tag{2}$$

$$cutsize(\Pi) = \sum_{n_j \in \mathcal{N}_E} (\lambda_j - 1). \tag{3}$$

Eq. 2, the cutsize is equal to the number of cut nets. In Eq. 3, each cut net n_j contributes $\lambda_j - 1$ to the cutsize. Hence, the hypergraph partitioning problem can be defined as the task of dividing a hypergraph into two or more parts such that the cutsize is minimized, while a given balance criterion (1) among the part weights is maintained. The hypergraph partitioning problem is known to be NP-hard [26].

3. COARSE-GRAIN DECOMPOSITION

Row and column coherences are important factors in sparse matrix partitioning for parallel SpMxV. Column coherence refers to the fact that nonzeroes on the same column incur the need for the same $\mathbf{x}$-vector entry. Row coherence refer to the fact that nonzeroes on the same row incur the contribution of scalar multiplication results to the same $\mathbf{y}$-vector entry. In a partitioning, disturbing column coherence incurs *expand* communication of $\mathbf{x}$-vector entries, whereas disturbing row coherency incurs *fold* communication of partial $\mathbf{y}$-vector results. Here, expand communication of an $\mathbf{x}$-vector entry x_j refers to the multicast operation where a processor, the one that owns the updated x_j value, sends (i.e., expands) the same x_j value to all processors that have nonzeros at column j. Fold communication of partial $\mathbf{y}$-vector results refers to the multinode accumulation operation on local, sparse $\mathbf{y}$ vectors computed by the processors. That is, every processor having at least one nonezero at row i generates a partial y_i result after local SpMxV computations, and these processors send their partial y_i results to the processor that owns y_i for a local accumulation operation to obtain the final y_i result. Note that 1D rowwise partitioning incurs only expand communication, because it respects row coherence by assigning entire rows to processors while disturbing column coherence. In a dual manner, 1D columnwise partitioning incurs only fold communication, because it respects column coherence by assigning entire columns to processors while disturbing row coherence. So in 1D matrix partitioning, the number of messages handled by a processor may be as high as $P-1$, for a parallel system with P processors. The worst-case total communication volume is $(P-1)m$ words for an $m \times m$ matrix, and this worst-case occurs when each row (column) stripe has at least one nonzero in each column (row) in rowwise (columnwise) partitioning. The standard graph-partitioning approach used for 1D matrix partitioning has some flaws such that the cutsize metric doesn't reflect the actual communication volume and it is restricted to symmetric matrices. Our previous hypergraph partitioning models [6, 7] correctly minimizes the communication volume in 1D matrix partitioning. Other recently proposed alternative models for 1D matrix partitioning were discussed in the excellent survey by Hendrickson and Kolda [17].

In a recent work [5, 9], we have investigated 2D fine-grain sparse-matrix partitioning which is overlooked in the literature. In this scheme, nonzeroes are allowed to be assigned individually to processors. Since neither row coherence nor column coherence is enforced, this scheme may incur both expand and fold operations so that the number of messages handled by a processor may be as high as $2(P-1)$. The worst-case communication volume is $2(P-1)m$ words in total. We proposed a fine-grain hypergraph-partitioning model [5, 9] which correctly minimizes total communication volume in this highly scalable 2D partitioning scheme. Experimental results showed that this 2D fine-grain partitioning model achieves drastic reduction in communication volume compared to 1D partitioning at the expense of increase in the number of total messages. This experimental finding is expected, because the 2D fine-grain model has a higher degree of freedom than the 1D models in minimizing communication volume since it does not enforce any coherence.

The 2D checkerboard partitioning scheme widely used for parallel SpMxV can be considered as a trade-off between 1D partitioning and 2D fine-grain partitioning schemes. This scheme respects both row and column coherences in a coarse level assuming a 2D processor mesh organization. It respects row coherence by assigning entire matrix rows to the processors in the same row of the processor mesh. It respects column coherence by assigning entire matrix columns to the processors in the same column of the processor mesh. Although both coherences are disturbed in processor level, the disturbances are confined to the processors of the same rows and columns. So, this scheme confines the expand operations to the columns and fold operations to the rows of the processor mesh. In this way, it reduces the maximum number of messages handled by a processor to $p+q-2$ for a $p \times q$ mesh of processors. The total communication volume may be as high as $(p+q-2)m$, and this worst-case occurs when each row and column of each submatrix has at least one nonzero. The 2D checkerboard partitioning schemes proposed in the literature [18, 27, 28, 33] mainly aim at load balancing and they do not exploit sparsity for reducing communication volume.

Here, we propose a novel hypergraph-partitioning approach for minimizing total communication volume while maintaining computational load balance in checkerboard partitioning. The proposed method is a two-phase method, in which each phase models either

the expand-communication volume or fold-communication volume. Therefore, we have two alternative schemes for the proposed approach. For the sake of simplicity in the presentation we will discuss only one scheme, the one which models the volume of expand communication in the first phase and then the volume of fold communication in the second phase. A dual discussion holds for the other scheme. We will discuss the checkerboard partitioning of an $m \times m$ square matrix A on a $p \times q$ processor mesh.

In the first phase, we perform a p-way 1D rowwise partitioning of matrix A using the column-net hypergraph-partitioning model proposed in our previous works [6, 7]. In the column-net hypergraph representation $\mathcal{H}_{\mathcal{R}} = (\mathcal{V}_{\mathcal{R}}, \mathcal{N}_{\mathcal{C}})$ of matrix A, there exist one vertex $v_i \in \mathcal{V}_{\mathcal{R}}$ and one net $n_j \in \mathcal{N}_{\mathcal{C}}$ for each row r_i and column c_j, respectively. Net $n_j \subseteq \mathcal{V}_{\mathcal{R}}$ contains the vertices corresponding to the rows which have a nonzero entry in column c_j. That is, $v_i \in n_j$ if and only if $a_{ij} \neq 0$. Weight w_i of a vertex $v_i \in \mathcal{V}_{\mathcal{R}}$ is set equal to the total number of nonzeroes in row r_i. As we discussed in [6, 7], by partitioning the column-net hypergraph into equally weighted vertex parts (maintaining balance condition in Eq. 1) so that nets are split among as few parts as possible (minimizing cutsize metric in Eq. 3), the model correctly minimizes total volume of expand communication while maintaining computational load balance in rowwise partitioning.

A p-way partition $\Pi_{\mathcal{R}} = \{\mathcal{V}_1, \ldots, \mathcal{V}_p\}$ of $\mathcal{H}_{\mathcal{R}}$ induces a p-way rowwise partitioning $\{\mathcal{R}_1, \ldots, \mathcal{R}_p\}$ of matrix A, where $\mathcal{R}_\alpha$ denotes the set of rows corresponding to $\mathcal{V}_\alpha$ for $\alpha = 1, 2, \ldots, p$. Without loss of generality we assume that the set $\mathcal{R}_\alpha$ of matrix rows is assigned to row α of processor mesh for $\alpha = 1, 2, \ldots, p$. This rowwise partitioning also corresponds to assigning a $m_\alpha \times m$ row-stripe A^r_α to row α of processor mesh, where $m_\alpha = |\mathcal{V}_\alpha|$. In a rowwise partitioning, a column is said to be internal if all of its nonzeroes are confined to a single row stripe. A column is said to be external if it has nonzeroes in the rows of at least two row stripes. An external column c_j which has nonzeroes in λ_j row stripes will incur the expand communication of x_j to $\lambda_j - 1$ processors along a column of the processor mesh if all nonzeroes of c_j are assigned to the same processor-column. Columnwise partitioning to be performed in the second phase satisfies this coherence.

In the second phase, we perform a q-way *multi-constraint* hypergraph partitioning on the row-net representation of matrix A for columnwise partitioning. In the row-net hypergraph representation $\mathcal{H}_{\mathcal{C}} = (\mathcal{V}_{\mathcal{C}}, \mathcal{N}_{\mathcal{R}})$, there exist one vertex $v_i \in \mathcal{V}_{\mathcal{C}}$ and one net $n_j \in \mathcal{N}_{\mathcal{R}}$ for each column c_i and row r_j, respectively. Net $n_j \subseteq \mathcal{V}_{\mathcal{C}}$ contains the vertices corresponding to the columns which have a nonzero entry in row r_j. That is, $v_i \in n_j$ if and only if $a_{ji} \neq 0$. As we discussed in [6, 7], minimizing the cutsize according to Eq. 3 corresponds to minimizing the total volume of fold communication of partial y_j results that need to be accumulated. A q-way partition $\Pi_{\mathcal{C}}$ of $\mathcal{H}_{\mathcal{C}}$ induces a q-way columnwise partitioning $\{\mathcal{C}_1, \ldots, \mathcal{C}_q\}$ of matrix A, where $\mathcal{C}_\beta$ denotes the set of columns assigned to processor column β, for $\beta = 1, 2, \ldots, q$. So, $(\Pi_{\mathcal{R}}, \Pi_{\mathcal{C}})$ determines a checkerboard partition, where $\mathcal{R}_\alpha$ and $\mathcal{C}_\beta$ denote the sets of rows and columns assigned to processor $P_{\alpha\beta}$ at row α and column β of the processor mesh. That is, processor $P_{\alpha\beta}$ will own nonzero $a_{ij} \neq 0$ of matrix A if $r_i \in \mathcal{R}_\alpha$ and $c_j \in \mathcal{C}_\beta$. For computational load balancing processors should be assigned roughly equal number of nonzeroes.

We introduce the multi-constraint hypergraph-partitioning concept for handling the load-balancing problem in our two-phase approach. The notion of multi-constraint and multi-objective partitioning has recently become popular in graph partitioning [21, 35] for the parallelization of multi-physics and multi-phase applications. In these applications, each constraint effectively corresponds to the computational load of the vertex in a different phase of the target parallel algorithm. Hence, maintaining balance on each constraint corresponds to maintaining load balance in each phase of the parallel algorithm. For our specific application, multiple weights of the vertices do not correspond to the weights of different phases. In fact they represent the computational loads that will be executed concurrently.

In our model, the rowwise partitioning $\Pi_{\mathcal{R}} = \{\mathcal{R}_1, \ldots, \mathcal{R}_p\}$ performed in the first phase already produces row stripes with roughly equal number of nonzeroes. In the second phase, each vertex v_i of $\mathcal{H}_{\mathcal{C}}$ is assigned p weights: $w_i(\alpha)$, for $\alpha = 1, 2, \ldots, p$. Here, $w_i(\alpha)$ is set equal to the number of nonzeroes of column c_i in row-stripe A^r_α. Note that all internal columns in the rowwise partitioning will have only one nonzero weight, whereas only external columns will have multiple nonzero weights. In any case, the sum of the p weights of each vertex will be equal to the total number of nonzeroes on the respective column. So, the balance constraint given in Eq. 1 is replaced with p balance constraints that should be maintained simultaneously, i.e.,

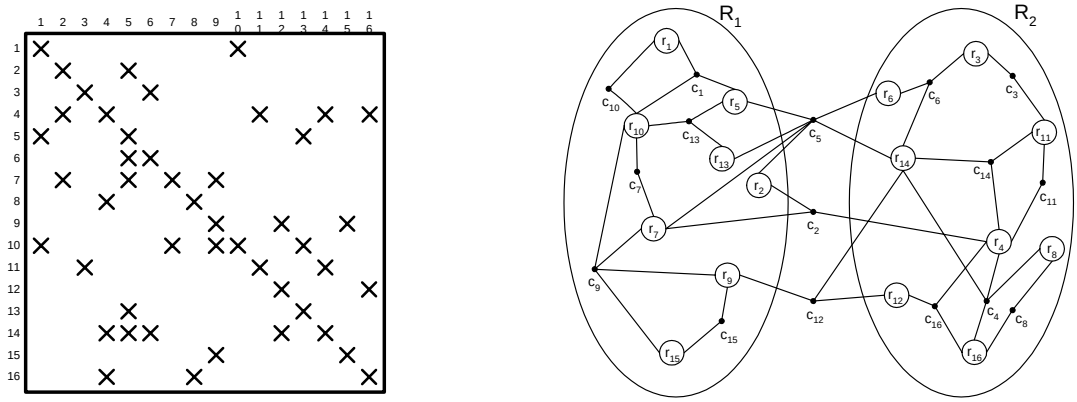
$$W_\beta(\alpha) \leq W_{avg}(1 + \varepsilon) \quad , \quad \beta = 1, 2, \ldots, q \quad \text{for each } \alpha = 1, 2, \ldots, p. \tag{4}$$

Here, $W_\beta(\alpha) = \sum_{v_i \in \mathcal{V}_\beta} w_i(\alpha)$ is the weight of part $\mathcal{V}_\beta$ of $\Pi_{\mathcal{C}}$ on the α-th constraint. Note that $W_\beta(\alpha)$ effectively denotes the number of nonzeroes assigned to processor $P_{\alpha\beta}$ in $(\Pi_{\mathcal{R}}, \Pi_{\mathcal{C}})$. Hence, in the second phase, maintaining the balance on each weight constraint during partitioning $\mathcal{H}_{\mathcal{C}}$ corresponds to maintaining computational load balance on the processors of each row of the 2D processor mesh.

Figures 1–3 illustrate the basic steps of our method for 2×2 checkerboard partitioning of a sample matrix. We labeled the vertices and nets of hypergraphs with letters "r" and "c" to denote row and column of a matrix, for simplicity in the presentation. In the hypergraph drawings, circles represent vertices, whereas dots represent nets. Figure 1(b) displays the column-net hypergraph representation $\mathcal{H}_{\mathcal{R}}$ of a sample matrix A given in Figure 1(a). It also shows a 2-way partition $\Pi_{\mathcal{R}}$ of $\mathcal{H}_{\mathcal{R}}$. Figure 2(b) shows the 2-way rowwise partitioning of sample matrix A induced by $\Pi_{\mathcal{R}}$. Figure 2(b) displays the row-net hypergraph representation $\mathcal{H}_{\mathcal{C}}$ of matrix A. It also shows a 2-way multi-constraint partition $\Pi_{\mathcal{C}}$ of $\mathcal{H}_{\mathcal{R}}$. In Fig. 2, $w_9(1) = 4$ and $w_9(2) = 0$ for internal column c_9 of row stripe $\mathcal{R}_1$, whereas $w_5(1) = 4$ and $w_5(2) = 2$ for external column c_5. Figure 3 displays the 2×2 checkerboard partitioning induced by $(\Pi_{\mathcal{R}}, \Pi_{\mathcal{C}})$.

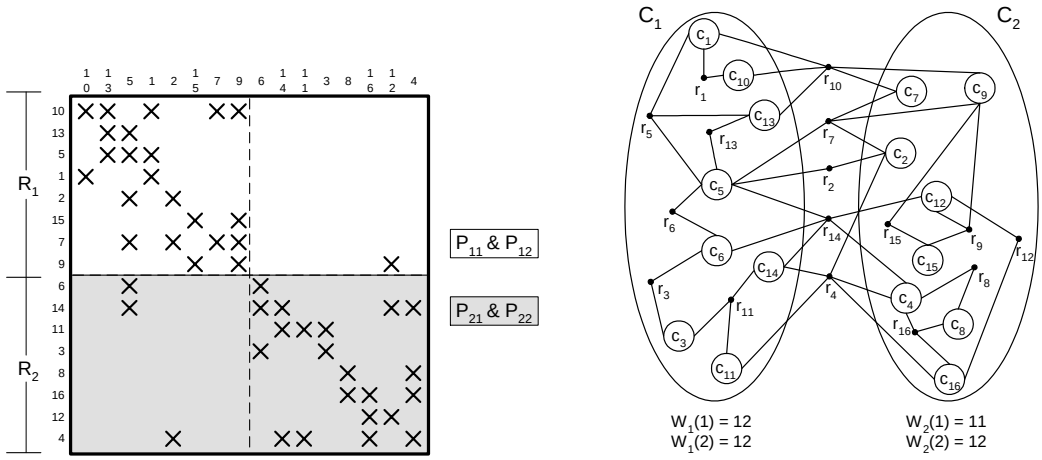
4. EXPERIMENTAL RESULTS

We have tested the validity of the proposed 2D partitioning method on various realistic sparse test matrices arising in different application domains [4, 10, 14, 15]. Table 1 illustrates the properties of the test matrices listed in the order of increasing number of nonzeros. The 2D partitioning results were obtained by running our multilevel hypergraph partitioning tool PaToH [8] on the hypergraphs. The 2D partitioning results were compared with the 1D partitionings obtained by running MeTiS [20] using the standard graph models, and PaToH using the 1D column/row-net hypergraph model presented in [6, 7]. For a specific P value, P-way partitioning of a test matrix constitutes a partitioning instance. For 2D partitioning instances, without loss of generality we selected $p = q = \sqrt{P}$. MeTiS and PaToH were run 50 times starting from different random seeds for each partitioning instance and average performance results are displayed in Figure 4. Communication volume values (in terms of the



(a) (b)

Figure 1: (a) A 16×16 sample nonsymmetric matrix A. (b) Phase 1: 2-way partitioning $\Pi_{\mathcal{R}}$ of column-net hypergraph representation $\mathcal{H}_{\mathcal{R}}$ of A.



(a) (b)

Figure 2: (a) Phase 1: 2-way rowwise partitioning of matrix A induced by $\Pi_{\mathcal{R}}$. (b) Phase 2: 2-way multi-constraint partitioning $\Pi_{\mathcal{C}}$ of row-net hypergraph representation $\mathcal{H}_{\mathcal{C}}$ of A.

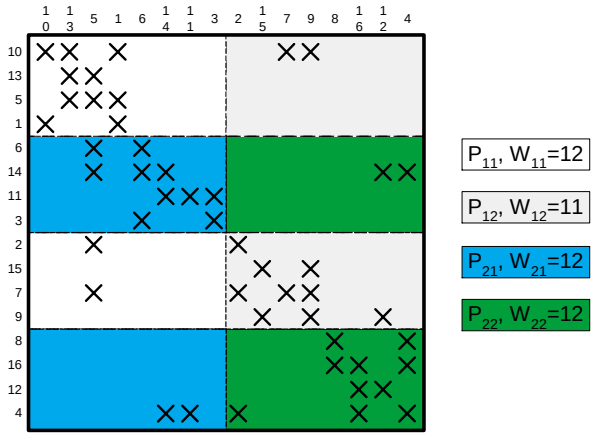


Figure 3: Coarse-grain 4-way partitioning of matrix A induced by $(\Pi_{\mathcal{R}}, \Pi_{\mathcal{C}})$.

Table 1: Properties of test matrices

name	number of rows/cols	number of nonzeros			
		total	per row/col		
			min	max	avg
sherman3	5005	20033	1	7	4.00
bcspwr10	5300	21842	2	14	4.12
ken-11	14694	82454	2	243	5.61
nl	7039	105089	1	361	14.93
ken-13	28632	161804	2	339	5.65
cq9	9278	221590	1	702	23.88
co9	10789	249205	1	707	23.10
pltexpA4-6	26894	269736	5	204	10.03
vibrobox	12328	342828	9	121	27.81
cre-d	8926	372266	1	845	41.71
cre-b	9648	398806	1	904	41.34
world	34506	582064	1	972	16.87
mod2	34774	604910	1	941	17.40
finan512	74752	615774	3	1449	8.24

number of words transmitted) are scaled by the number of rows/columns of the respective test matrices. Average and maximum number of messages handled by a single processor are also displayed in this figure. The percent load imbalance values are below 3% for all partitioning results displayed in these figures, where percent imbalance ratio is defined as $100 \times (W_{max} - W_{avg})/W_{avg}$. Figure 4(d) displays the partitioning times on a workstation equipped with a 133 MHz PowerPC processor with 64 Mbytes of memory.

In terms of total communication volume, Figure 4(a), the proposed method produces better partitions than the standard graph model on 29 instances out of 42 instances. On the average, the proposed method produces 23%, 25% and 27% better partitions than the standard graph model on 16, 32 and 64 processors. The proposed method produces comparable results with the 1D hypergraph model in terms of communication volume.

In the proposed method, the upper bound on the maximum number of messages handled by a processor is 6, 10, and 14 for a parallel system with 16, 32, and 64 processors, respectively. As seen in Figure 4(b), these upper bound values are reached on 31 instances out of 42 instances. In terms of maximum number of messages handled by a single processor, the proposed method produces drastically better partitions than the 1D graph and hypergraph models on every instance. On the average, it produces 56%, 62% and 72% better partitions than the 1D graph model on 16, 32 and 64 processors. These relative performance figures slightly reduce to 55%, 60% and 69%, respectively, for the 1D hypergraph model. As expected the performance gap increases rapidly with increasing number of processors in favor of the proposed method.

In terms of average number of messages handled by a single processor, Figure 4(c), the proposed method produces better partitions than the 1D graph model on every instance except 32- and 64- way partitionings of BCSPWR10 matrix. It produces better partitions than the 1D hypergraph model on 33 instances out of 42 instances. On the average, the proposed coarse-grain method produces 54%, 57%, and 62% better partitions than the 1D graph model on 16, 32, and 64 processors. These relative performance figures considerably reduce to 44%, 42% and 46%, respectively, for the 1D hypergraph model. Note that these relative performance figures also show the relative figures on the total number of messages.

The proposed method is approximately 3 times slower than the standard graph model, on the overall average (Figure 4(d)). The execution times of 1D hypergraph model and coarse-grain method are almost the same. For P-way partitioning, 1D hypergraph model necessitates P-way partitioning of the hypergraph representation of the sparse matrix. However, the proposed method requires two $\sqrt{P}$-way partitionings of the hypergraph representations of the sparse matrix. Since the current PaToH implementations achieve multi-way partitioning by recursive partitioning, two $\sqrt{P}$-way partitionings is usually faster than one P-way partitioning, even though one of the two $\sqrt{P}$-way partitionings is a multi-constraint partitioning.

5. CONCLUSION

A coarse-grain decomposition method was proposed for 2D partitioning of sparse matrices which has a better upper bound on the maximum number of messages handled by a single processor than the existing methods. The proposed model reduces the 2D matrix partitioning problem to a multi-constraint hypergraph partitioning problem so that partitioning objectives correspond to minimizing communication volume while maintaining load balance during repeated matrix-vector multiplication. The performance of the proposed 2D partitioning model was tested against 1D partitioning through graph and hypergraph models on a wide range of realistic sparse matrices. On the average, the 2D partitionings achieved about 59 and 44 percent better decompositions than 1D graph and hypergraph models, respectively, in the number of communications required for a single parallel matrix-vector multiplication.

6. REFERENCES

[1] A. Afework, M. D. Beynon, F. Bustamante, A. Demarzo, R. Ferreira, R. Miller, M. Silberman, J. Saltz, A. Sussman, and H. Tsang. Digital dynamic telepathology - the Virtual Microscope. In *Proceedings of the 1998 AMIA Annual Fall Symposium*. American Medical Informatics Association, Nov. 1998.

[2] T. Bultan and C. Aykanat. A new mapping heuristic based on mean field annealing. *Journal of Parallel and Distributed Computing*, 16:292–305, 1992.

[3] W. Camp, S. J. Plimpton, B. Hendrickson, and R. W. Leland. Massively parallel methods for engineering and science problems. *Communication of ACM*, 37(4):31–41, April 1994.

[4] W. J. Carolan, J. E. Hill, J. L. Kennington, S. Niemi, and S. J. Wichmann. An empirical evaluation of the korbx algorithms for military airlift applications. *Operations Research*, 38(2):240–248, 1990.

[5] U. V. Çatalyürek. *Hypergraph Models for Sparse Matrix Partitioning and Reordering*. PhD thesis, Bilkent University, Computer Engineering and Information Science, Nov 1999.

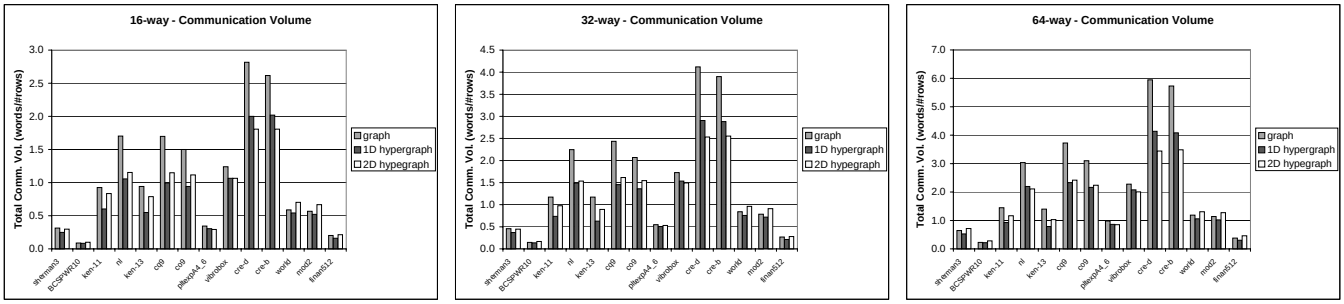
[6] U. V. Çatalyürek and C. Aykanat. Decomposing irregularly sparse matrices for parallel matrix-vector multiplications. *Lecture Notes in Computer Science*, 1117:75–86, 1996.

[7] U. V. Çatalyürek and C. Aykanat. Hypergraph-partitioning based decomposition for parallel sparse-matrix vector multiplication. *IEEE Transactions on Parallel and Distributed Systems*, 10(7):673–693, 1999.

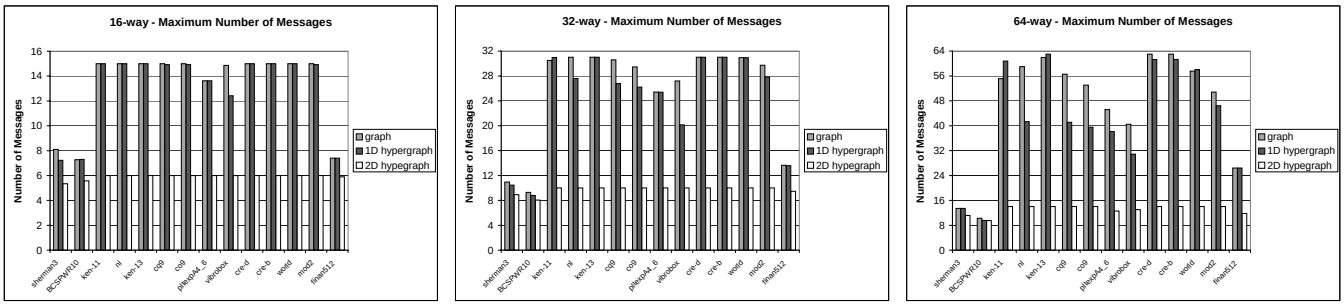
[8] U. V. Çatalyürek and C. Aykanat. *PaToH: A Multilevel Hypergraph Partitioning Tool, Version 3.0*. Bilkent University, Department of Computer Engineering, Ankara, 06533 Turkey, 1999.

[9] U. V. Çatalyürek and C. Aykanat. A fine-grain hypergraph model for 2D decomposition of sparse matrices. In *Proceedings of 15th International Parallel and Distributed Processing Symposium (IPDPS)*, San Francisco, CA, April 2001.

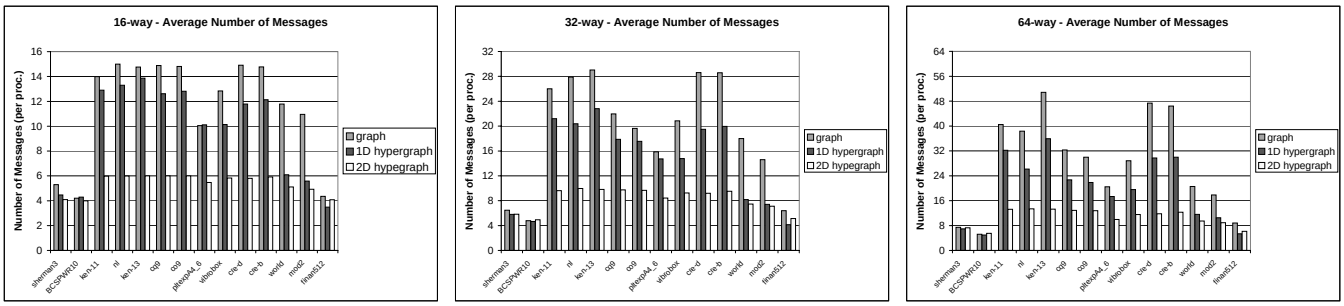
[10] I. O. Center. Linear programming problems. ftp://col.biz.uiowa.edu:pub/testprob/lp/gondzio.



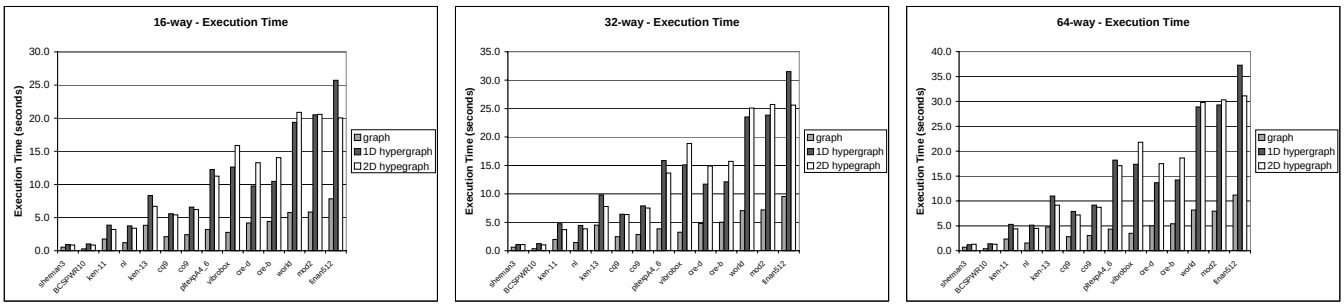
(a) Total communication volume scaled by #rows/columns



(b) Maximum number of messages handled by a single processor



(c) Average number of messages handled by a processor



(d) Execution times of decomposition heuristics

Figure 4: Average communication requirements of the proposed coarse-grain method for 2D partitioning and the existing 1D partitioning models.

[11] C. F. Cerco and T. Cole. User's guide to the CE-QUAL-ICM three-dimensional eutrophication model, release version 1.0. Technical Report EL-95-15, US Army Corps of Engineers Water Experiment Station, Vicksburg, MS, 1995.

[12] C. Chang, A. Acharya, A. Sussman, and J. Saltz. T2: A customizable parallel database for multi-dimensional data. *ACM SIGMOD Record*, 27(1):58–66, Mar. 1998.

[13] C. Chang, T. Kurc, A. Sussman, U. V. Çatalyürek, and J. Saltz. A hypergraph-based workload partitioning strategy for parallel data aggregation. In *Proceedings of the Eleventh SIAM Conference on Parallel Processing for Scientific Computing*. SIAM, Mar. 2001.

[14] T. Davis. University of florida sparse matrix collection: http://www.cise.ufl.edu/ davis/sparse/. *NA Digest*, 92/96/97(42/28/23), 1994/1996/1997.

[15] I. S. Duff, R. Grimes, and J. Lewis. Sparse matrix test problems. *ACM Transactions on Mathematical Software*, 15(1):1–14, march 1989.

[16] B. Hendrickson. Graph partitioning and parallel solvers: has the emperor no clothes? *Lecture Notes in Computer Science*, 1457:218–225, 1998.

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

[18] B. Hendrickson, R. Leland, and S. Plimpton. An efficient parallel algorithm for matrix-vector multiplication. *Int. J. High Speed Computing*, 7(1):73–88, 1995.

[19] M. Kaddoura, C. W. Qu, and S. Ranka. Partitioning unstructured computational graphs for nonuniform and adaptive environments. *IEEE Parallel and Distributed Technology*, 3(3):63–69, 1995.

[20] G. Karypis and V. Kumar. *MeTiS A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices Version 4.0*. University of Minnesota, Department of Comp. Sci. and Eng., Army HPC Research Center, Minneapolis, 1998.

[21] G. Karypis and V. Kumar. Multilevel algorithms for multi-constraint graph partitioning. Technical Report 98-019, University of Minnesota, Department of Computer Science/Army HPC Research Center, Minneapolis, MN 55455, May 1998.

[22] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, to appear.

[23] V. Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to Parallel Computing: Design and Analysis of Algorithms*. Benjamin/Cummings Publishing Company, Redwood City, CA, 1994.

[24] T. Kurc, C. Chang, R. Ferreira, A. Sussman, and J. Saltz. Querying very large multi-dimensional datasets in ADR. In *Proceedings of the 1999 ACM/IEEE SC99 Conference*. ACM Press, Nov. 1999.

[25] V. Lakamsani, L. N. Bhuyan, and D. S. Linthicum. Mapping molecular dynamics computations on to hypercubes. *Parallel Computing*, 21:993–1013, 1995.

[26] T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout*. Willey–Teubner, Chichester, U.K., 199.

[27] J. G. Lewis, D. G. Payne, and R. A. van de Geijn. Matrix-vector multiplication and conjugate gradient algorithms on distributed memory computers. In *Proceedings of the Scalable High Performance Computing Conference*, 1994.

[28] J. G. Lewis and R. A. van de Geijn. Distributed memory matrix-vector multiplication and conjugate gradient algorithms. In *Proceedings of Supercomputing'93*, pages 15–19, Portland, OR, November 1993.

[29] R. A. Luettich, J. J. Westerink, and N. W. Scheffner. *ADCIRC*: An advanced three-dimensional circulation model for shelves, coasts, and estuaries. Technical Report 1, Department of the Army, U.S. Army Corps of Engineers, Washington, D.C. 20314-1000, December 1991.

[30] O. C. Martin and S. W. Otto. Partitioning of unstructured meshes for load balancing. *Concurrency: Practice and Experience*, 7(4):303–314, 1995.

[31] The Moderate Resolution Imaging Spectrometer. http://ltpwww.gsfc.nasa.gov/MODIS/MODIS.html.

[32] NASA Goddard Distributed Active Archive Center (DAAC). Advanced Very High Resolution Radiometer Global Area Coverage (AVHRR GAC) data. http://daac.gsfc.nasa.gov/CAMPAIGN_DOCS/LAND_BIO/origins.html.

[33] A. T. Ogielski and W. Aielo. Sparse matrix computations on parallel processor arrays. *SIAM Journal on Numerical Analysis*, 1993.

[34] C.-W. Qu and S. Ranka. Parallel incremental graph partitioning. *IEEE Transactions on Parallel and Distributed Systems*, 8(8):884–896, 1997.

[35] K. Schloegel, G. Karypis, and V. Kumar. A new algorithm for multi-objective graph partitioning. Technical Report 99-003, University of Minnesota, Department of Computer Science/Army HPC Research Center, Minneapolis, MN 55455, Sep 1999.

[36] T. Tanaka. Configurations of the solar wind flow and magnetic field around the planets with no magnetic field: calculation by a new MHD. *Jounal of Geophysical Research*, 98(A10):17251–62, Oct 1993.

[37] U.S. Geological Survey. Land satellite (LANDSAT) thematic mapper (TM). http://edcwww.cr.usgs.gov/nsdi/html/landsat_tm/landsat_tm.