# Decomposing Irregularly Sparse Matrices for Parallel Matrix-Vector Multiplication[*]

Ümit V. Çatalyürek and Cevdet Aykanat
Computer Engineering Department, Bilkent University
06533 Bilkent, Ankara, Turkey

**Abstract.** In this work, we show the deficiencies of the graph model for decomposing sparse matrices for parallel matrix-vector multiplication. Then, we propose two hypergraph models which avoid all deficiencies of the graph model. The proposed models reduce the decomposition problem to the well-known hypergraph partitioning problem widely encountered in circuit partitioning in VLSI. We have implemented fast Kernighan-Lin based graph and hypergraph partitioning heuristics and used the successful multilevel graph partitioning tool (Metis) for the experimental evaluation of the validity of the proposed hypergraph models. We have also developed a multilevel hypergraph partitioning heuristic for experimenting the performance of the multilevel approach on hypergraph partitioning. Experimental results on sparse matrices, selected from Harwell-Boeing collection and NETLIB suite, confirm both the validity of our proposed hypergraph models and appropriateness of the multilevel approach to hypergraph partitioning.

## 1 Introduction

Iterative solvers are widely used for the solution of large, sparse, linear system of equations on multicomputers. Three basic types of operations are repeatedly performed at each iteration. These are linear operations on dense vectors, inner product(s) of dense vectors, and sparse-matrix vector product of the form $\mathbf{y} = \mathbf{A}\mathbf{x}$, where $\mathbf{y}$ and $\mathbf{x}$ are dense vectors, and $\mathbf{A}$ is a matrix with the same sparsity structure as the coefficient matrix [5, 12]. All of these basic operations can be performed concurrently by distributing either the rows or the columns of the matrix $\mathbf{A}$ and the components of the dense vectors in the same way. These two decomposition schemes are referred here as *rowwise* and *columnwise* decomposition schemes, respectively. Note that these two decomposition schemes are one-dimensional decomposition of matrix $\mathbf{A}$ which is a two-dimensional data structure. Both of these two decomposition schemes induce a computational distribution such that each processor is held responsible for updating the values of those vector components assigned to itself. With this data distribution scheme, linear vector operations and inner-product operations can be easily and efficiently parallelized by an even distribution of vector components to processors [5, 12]. Linear vector operations do not necessitate communication, whereas

[*] This work is partially supported by the Commission of the European Communities, Directorate General for Industry under contract ITDC 204-82166, and Turkish Science and Research Council under grant EEEAG-160.

inner-product operations introduce global communication overhead which does not scale up with increasing problem size.

Sparse-matrix vector product computations constitute the most time consuming operation in iterative solvers. In parallel matrix-vector multiplication, rowwise and columnwise decomposition schemes necessitate communication just before and after the local matrix-vector product computations, respectively. Hence, these two schemes can also be considered as the pre and post communication schemes, respectively. In rowwise decomposition scheme, processors need some nonlocal components of the global **x**-vector, depending on the sparsity pattern of their local rows, just before the local matrix-vector product computations. Each processor send some of its local **x**-vector components to those processor(s) which need them. After receiving the needed nonlocal **x** components, each processor can concurrently compute its local components of the global **y**-vector by performing a local matrix-vector product. In columnwise decomposition scheme, after local matrix-vector product computations, processors send the non-local components of their computed **y**-vectors to those processor(s) which need them, depending on the sparsity pattern of their local columns. After receiving the needed **y** components, each processor can concurrently complete the computation of its local **y**-vector by simply adding these received values to its appropriate local **y**-vector locations. Hence, by weighting each row or column by its nonzero entry count, load balancing problem can be considered as the *number partitioning* problem. However, different row or column partitionings with good load balance may also significantly differ the communication requirement. Unfortunately, the communication requirement scales up with increasing problem size. The minimization of the communication overhead while maintaining the computational load balance reduces to the *domain decomposition problem*, where the sparse matrix **A** constitutes the domain of problem.

Almost all domain decomposition methods proposed in the literature employ *graph model* [9, 13]. In this work, we show the deficiencies of the graph model for decomposing sparse matrices for parallel matrix vector multiplication. The first deficiency is that it can only be used for symmetric square matrices. The second deficiency is the fact that the graph model does not reflect the actual communication requirement which will be described in Section 2.3. In this work, we propose two *hypergraph models* which avoid all deficiencies of the graph model. The proposed models enable the representation and hence the decomposition of unsymmetric square and rectangular matrices as well as symmetric matrices. Furthermore, they introduce a much more accurate representation for the communication requirement. The proposed models reduce the decomposition problem to the well-known *hypergraph partitioning* problem widely encountered in circuit partitioning in VLSI layout design. Hence, the proposed models will be amenable to the advances in the circuit partitioning heuristics and tools to be developed in VLSI community.

Domain decomposition is a preprocessing introduced for the sake of efficient parallelization of the given problem. Hence, heuristics used for decomposition should run in low order polynomial time. Kernighan-Lin (KL) based heuristics are widely used for graph and hypergraph partitioning because of their short run-times and good quality results. Therefore, we selected and implemented fast $k$-way KL-based graph and hypergraph partitioning heuristics for experimenting the validity of our proposed hypergraph models. Here, $k$ represents the number of processors on the target multicomputer. Recently, multilevel graph partitioning heuristics are proposed leading to successful graph partitioning tools Chaco [8] and Metis [10]. We have also exploited the multilevel partitioning methods for the experimental verification of our proposed hypergraph models in two approaches. In the first approach, Metis graph partitioning tool is used as a black box by transforming hypergraphs to graphs using the traditional clique-net model. In the second approach, lack of existence of multilevel hypergraph partitioning tool led us to develop a multilevel hypergraph partitioning heuristic, for fair comparison of two models.

## 2 Graph Model and Its Deficiencies

In this section, we discuss the deficiencies of graph model for decomposing sparse matrices for parallel matrix-vector multiplication.

### 2.1 Graphs and Graph Partitioning Problem

An undirected graph $\mathcal{G}=(\mathcal{V},\mathcal{E})$ is defined as a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$. Every edge $e_{ij}\in\mathcal{E}$ connects a pair of vertices $v_i$ and $v_j$. The degree $d_i$ of a vertex $v_i$ is equal to the number of edges incident to $v_i$. Let $w_i$ and $c_{ij}$ denote the weight of vertex $v_i\in\mathcal{V}$ and the cost of edge $e_{ij}\in\mathcal{E}$, respectively.

$\Pi=(P_1,\ldots,P_k)$ is a *k-way partition* of $\mathcal{G}$ if the following conditions hold: each part $P_\ell, 1\leq\ell\leq k$, is a nonempty subset of $\mathcal{V}$, parts are pairwise disjoint ($P_i\cap P_j=\emptyset$ for all $1\leq i<j\leq k$), and union of $k$ parts is equal to $\mathcal{V}$. In a partition $\Pi$ of $\mathcal{G}$, an edge is said to be *cut* if its pair of vertices belong to two different parts, and otherwise *uncut*. The set of cut (*external*) edges for a partition $\Pi$ are denoted as $\mathcal{E}_E$. The *cutsize* definition for representing the cost $\chi(\Pi)$ of a partition $\Pi$ is

$$\chi(\Pi)=\sum_{e_{ij}\in\mathcal{E}_E} c_{ij} \qquad (1)$$

In (1), each cut edge $e_{ij}$ contributes its cost $c_{ij}$ to the cutsize. In a partition $\Pi$ of $\mathcal{G}$, the size of a part is defined as the sum of the weights of the vertices in that part. Hence, graph partitioning problem can be defined as the task of dividing a graph into two or more parts such that the cutsize is minimized, while a given balance criterion among the part sizes is maintained.

### 2.2 Graph Model for Decomposition

Graph representation of only structurally symmetric matrices will be discussed in the decomposition context, since graph model is restricted to symmetric matrices. A symmetric sparse matrix $\mathbf{A}$ can be represented as an undirected graph $\mathcal{G}_A=(\mathcal{V},\mathcal{E})$. The vertices in the vertex set $\mathcal{V}$ correspond to the rows/columns of the matrix $\mathbf{A}$. In the pre-communication scheme, each vertex $v_i\in\mathcal{V}$ corresponds to the atomic task $i$ of computing the inner product of row $i$ with the column vector $\mathbf{x}$. In the post-communication scheme, each vertex $v_i \in \mathcal{V}$ corresponds to the atomic task $i$ of computing the sparse SAXPY/DAXPY operation $\mathbf{y} = \mathbf{y} + x_i \mathbf{a}_{*i}$, where $\mathbf{a}_{*i}$ denotes the $i$-$th$ column of matrix $\mathbf{A}$. Hence, in both pre and post communication schemes, each nonzero entry in a row and column of $\mathbf{A}$ incurs a multiply-and-add operation during the local matrix-vector product computations in the pre and post communication schemes, respectively. Thus, computational load $w_i$ of mapping row/column $i$ to a processor is the number of nonzero entries in row/column $i$.

In the edge set $\mathcal{E}$, $e_{ij} \in \mathcal{E}$ if and only if $a_{ij}$ and $a_{ji}$ of matrix $\mathbf{A}$ are nonzeros. Hence, the vertices in the adjacency list of a vertex $v_i$ denote the column (row) indices of the off-diagonal nonzeros in row $i$ (column $i$) of $\mathbf{A}$. In the pre-communication scheme, each edge $e_{ij} \in \mathcal{E}$ corresponds to the exchange of updated $x_i$ and $x_j$ values between the atomic tasks $i$ and $j$, just before the local matrix-vector product computations. In the post-communication scheme, each edge $e_{ij} \in \mathcal{E}$ corresponds to the exchange of partial $y_i$ and $y_j$ results between the atomic tasks $i$ and $j$, just after the local matrix-vector product computations. In both schemes, each edge represents the bidirectional interaction between the respective pair of vertices. Hence, by setting $c_{ij} = 2$ for each edge $e_{ij} \in \mathcal{E}$, both rowwise and columnwise decomposition of matrix $\mathbf{A}$ reduces to the $k$-way partitioning of its associated graph $\mathcal{G}_A$ according to the cutsize definition given in (1). Thus, minimizing the cutsize according to (1) corresponds to the goal of minimizing the total volume of interprocessor communication. Maintaining the balance among part sizes corresponds to maintaining the computational load balance during local matrix-vector product computations.

### 2.3 Deficiencies of the Graph Model

As mentioned earlier, graph model is restricted to representing structurally symmetric matrices. Furthermore, the graph model does not reflect the actual communication requirement. Graph model treats all cut edges in an identical manner while computing the cutsize (i.e., 2 words per cut edge). However, $r$ cut edges stemming from a vertex $v_i$ in part $P_\ell$ to $r$ vertices $v_{i_1}, v_{i_2}, \ldots, v_{i_r}$ in part $P_m$ incur only $r{+}1$ communications instead of $2r$ in both pre and post communication schemes. Because, in the pre-communication (post-communication) scheme, $P_\ell$ ($P_m$) sends $x_i$ ($y_{i_1}, y_{i_2}, \ldots, y_{i_r}$) to processor $P_m$ ($P_\ell$) while $P_m$ ($P_\ell$) sends $x_{i_1}, x_{i_2}, \ldots, x_{i_r}$ ($y_i$) to processor $P_\ell$ ($P_m$).

## 3 Hypergraph Models for Decomposition

In this section, we propose two *hypergraph models* for mapping sparse-matrix vector multiplication which avoids the deficiencies of the graph model.

### 3.1 Hypergraphs and Hypergraph Partitioning Problem

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., $n_j \subseteq \mathcal{V}$. Vertices in a net $n_j$ are called its *pins* and denoted as $pins[n_j]$. The size of a net is equal to the number of its pins, i.e., $s_j = |nets[n_j]|$. The set of nets connected to a vertex $v_i$ is denoted as $nets[v_i]$. The degree of a vertex is equal to the number of nets it is connected to, i.e., $d_i = |nets[v_i]|$. Let $w_i$ and $c_j$ denote the weight of vertex $v_i \in \mathcal{V}$ and the cost of net $n_j \in \mathcal{N}$, respectively.

Definition of $k$-way partition of hypergraphs is identical to that of graphs. In a partition $\Pi$ of $\mathcal{H}$, a net that has at least one pin (vertex) in a part is said to *connect* that part. Let $\delta_j$ denotes the number of parts connected by net $n_j$. A net $n_j$ is said to be *cut* if it connects more than one part (i.e., $\delta_j > 1$), and *uncut* (i.e., $\delta_j = 1$) otherwise. The set of cut (*external*) nets for a partition $\Pi$ are denoted as $\mathcal{N}_E$. There are various *cutsize* definitions for representing the cost $\chi(\Pi)$ of a partition $\Pi$. Two relevant definitions are:

$$(a) \quad \chi(\Pi) = \sum_{n_j \in \mathcal{N}_E} c_j \qquad and \qquad (b) \quad \chi(\Pi) = \sum_{n_j \in \mathcal{N}_E} c_j(\delta_j - 1). \qquad (2)$$

In (2.a), cutsize is equal to the sum of the costs of the cut nets. In (2.b), each cut net $n_j$ contributes $c_j(\delta_j - 1)$ to the cutsize. Hence, 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 among the part sizes is maintained.

## 3.2 Two Hypergraph Models for Decomposition

We propose two hypergraph models for the decomposition. These models are referred to here as the *column-net* and *row-net* models. In the column-net model, matrix $\mathbf{A}$ is represented as the hypergraph $\mathcal{H}_\mathcal{C}(\mathcal{V}_\mathcal{R}, \mathcal{N}_\mathcal{C})$. The vertex and net sets $\mathcal{V}_\mathcal{R}$ and $\mathcal{N}_\mathcal{C}$ correspond to the rows and columns of matrix $\mathbf{A}$, respectively. There exist one vertex $v_i$ and one net $n_j$ for each row $i$ and column $j$, respectively. Net $n_j$ contains the vertices corresponding to the rows which have a nonzero entry on column $j$. That is, $v_i \in n_j$ if and only if $a_{ij} \neq 0$. Each vertex $v_i \in \mathcal{V}_\mathcal{R}$ corresponds to the atomic task $i$ of computing the inner product of row $i$ with the column vector $\mathbf{x}$. Hence, the weight $w_i = d_i$ is associated with each vertex $v_i \in \mathcal{V}_\mathcal{R}$. Nets of $\mathcal{H}_\mathcal{C}$ represent the *dependency* relations of the atomic tasks to the $\mathbf{x}$-vector components in the pre-communication scheme. That is, each net $n_j \subseteq \mathcal{V}_\mathcal{R}$ denotes the set of atomic tasks that need $x_j$.

The row-net model can be considered as the dual of the column-net model. In this model, matrix $\mathbf{A}$ is represented as the hypergraph $\mathcal{H}_\mathcal{R}(\mathcal{V}_\mathcal{C}, \mathcal{N}_\mathcal{R})$. The vertex and net sets $\mathcal{V}_\mathcal{C}$ and $\mathcal{N}_\mathcal{R}$ correspond to the columns and rows of the matrix $\mathbf{A}$, respectively. There exist one vertex $v_i$ and one net $n_j$ for each column $i$ and row $j$, respectively. Net $n_j$ contains the vertices corresponding to the columns which have a nonzero entry on row $j$. That is, $v_i \in n_j$ if and only if $a_{ji} \neq 0$. Each vertex $v_i \in \mathcal{V}_\mathcal{C}$ corresponds to the atomic task $i$ of computing the sparse SAXPY/DAXPY operation $\mathbf{y} = \mathbf{y} + x_i \mathbf{a}_{*i}$. Hence, the weight $w_i = d_i$ is associated with each vertex $v_i \in \mathcal{V}_\mathcal{C}$. Nets of $\mathcal{H}_\mathcal{R}$ represent the *dependency* relations of the computation of $\mathbf{y}$-vector components to the atomic tasks represented by vertices of $\mathcal{H}_\mathcal{R}$ in the post-communication scheme. That is, each net $n_j \subseteq \mathcal{V}_\mathcal{C}$ denotes the set of atomic task results needed to compute $y_j$.

By assigning unit costs to the nets (i.e., $c_j = 1$ for each net $n_j$), the proposed column-net and row-net models reduce the decomposition problem into $k$-way hypergraph partitioning problem according to the cutsize definition given in (2.b) for the pre and post communication schemes, respectively. Part size definition is identical to that of the graph model. Assume that part $P_\ell$ is assigned to processor $\ell$. Let $\mathcal{C}[i]$ denotes the *connectivity* set of net $n_i$ which is defined as the set of parts (processors) connected by the net $n_i$. Note that $\delta_i = |\mathcal{C}[i]|$. In the column-net model together with the pre-communication scheme, a cut net $n_i$ indicates that processor $part[x_i] \in \mathcal{C}[i]$ should send its local $x_i$ to those processors in the connectivity set of net $n_i$ except itself (i.e., to processors in the set $\mathcal{C}[i] - \{part[x_i]\}$. Hence, processor $part[x_i]$ should send its local $x_i$ to $\delta_i - 1$ distinct processors. Here, $part[x_i]$ denotes the part (processor) assignment for $x_i$. In the row-net model together with the post-communication scheme, a cut net $n_i$ indicates that processor $part[y_i] \in \mathcal{C}[i]$ should receive the partial $y_i$ results from those processors in the connectivity set of net $n_i$ except itself (i.e., from processors in the set $\mathcal{C}[i] - \{part[y_i]\}$). Hence, processor $part[y_i]$ should receive partial $y_i$ results from $\delta_i - 1$ distinct processors. Thus, in column-net and row-net models, minimizing the cutsize according to (2.b) corresponds to minimizing the actual volume of interprocessor communication during pre and post communication phases, respectively. Maintaining the balance among part sizes corresponds to maintaining the computational load balance during local matrix-vector product computations. Note that row-net and column-net models become identical in symmetric square matrices.

Figure 1 illustrates 4-way graph and hypergraph partitions corresponding to the partial decomposition of a symmetric matrix. Here, assume that part $P_\ell$ is assigned to processor $\ell$ for $\ell = 1, 2, 3, 4$. As seen in Fig. 1(a), cutsize in the graph model is $2 \times 5 = 10$ since there are 5 cut edges. However, actual volume of communication is 7 in both pre and post communication schemes. For example, in the pre-communication scheme, processor 1 should send $x_1$ to both processors 2 and 4 only once, whereas processors 2 and 4 should send 3 and 2 local $x_i$ values to processor 1, respectively. As seen in Fig. 1(b), each cut-net $n_i$, for $i = 2, 3, \ldots, 6$, contributes 1 to the cutsize since $\delta_2 = \delta_3 = \ldots = \delta_6 = 2$, and cut-net $n_1$ contributes 2 to the cutsize since $\delta_1 = 3$. Hence, the cutsize in the hypergraph model is 7 thus leading to an accurate modeling of the communication requirement.

## 4 Decomposition Heuristics

Kernighan-Lin (KL) based heuristics are widely used for graph and hypergraph partitioning because of their short run-times, and good quality results. KL algorithm is an iterative improvement heuristic originally proposed for 2-way graph partitioning (bipartitioning) [11]. This algorithm became the basis for most of the subsequent partitioning algorithms, all of which we call the KL-based algorithms. KL algorithm performs a number of passes until it finds a locally minimum partition. Each pass consists of a sequence of vertex swaps. The same swap strategy was applied to hypergraph partitioning problem by Schweikert-Kernighan [14]. Fiduccia-Mattheyses (FM) [6] introduced a faster implementation of KL algorithm for hypergraph partitioning. They proposed vertex move concept instead of vertex swap. This modification as well as proper data structures, e.g., bucket

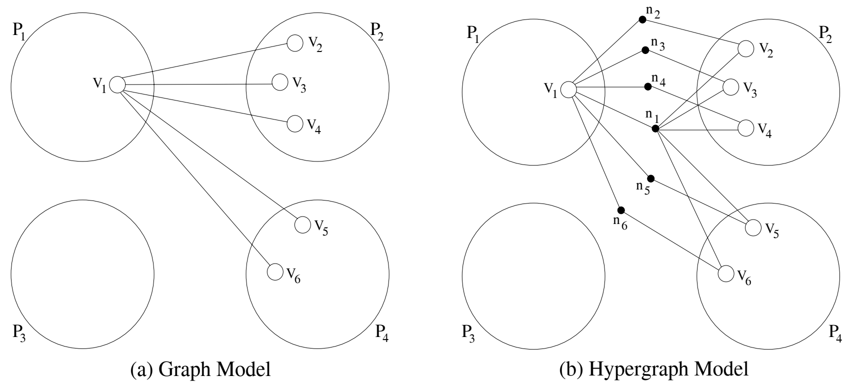

**Fig. 1.** A partial 4-way decomposition of a symmetric matrix in (a) graph, (b) hypergraph models

lists, reduced the time complexity of a single pass of KL algorithm to linear in the size of the graph and the hypergraph. Here, *size* refers to the number of edges and pins in a graph and hypergraph, respectively. In this work, we have implemented $k$-way FM-based graph and hypergraph partitioning heuristics for experimenting the validity of our proposed hypergraph models.

The performance of FM deteriorates for large and/or too sparse and/or dense graphs/hypergraphs. Many clustering algorithms have been proposed especially for hypergraphs to alleviate this problem [2]. Clustering corresponds to coalescing highly interacting vertices to supernodes as a preprocessing to FM. Recently, *multilevel* graph partitioning methods have been proposed leading to successful graph partitioning tools Chaco [8] and Metis [10]. These multilevel heuristics consists of 3 phases, namely *coarsening*, *initial partitioning*, and *uncoarsening*. In the first phase, multilevel clustering is successively applied starting from the original graph by adopting various matching heuristics until number of vertices in the coarsened graph reduces below a predetermined threshold value. In the second phase, coarsest graph is partitioned using various heuristics including FM. In the third phase, partition found in the second phase is successively projected back towards the original graph by refining the projected partitions on intermediate level uncoarser graphs using various heuristics including FM.

### 4.1 Clique-Net Model for Graph Representation of Hypergraphs

The goal in this approach is to exploit the Metis graph partitioning tool as a black box. So, we use the traditional clique-net model to transform hypergraphs to graphs. In this transformation model, the vertex set of the target graph is equal to the vertex set of the given hypergraph. Each net of the given hypergraph is represented by a clique of vertices corresponding to its pins. Vertex weights of the hypergraph become the vertex weights of the graph. Costs of the edges are equal to the sum of the costs of the nets that they represent. If an edge is in the cut set of a graph partitioning then all nets represented by this edge are in the cut set of hypergraph partitioning and vice versa.

The deficiency of this graph model is that it treats a net with size $s$ in the hypergraph as $s(s-1)/2$ edges. This strategy exaggerates the importance of the nets that have more than two terminals and the exaggeration grows with the square of the size of the net [14]. In our current implementation, we remove all nets of size larger than $T$ during the transformation. Furthermore, for each net $n_j$, we select $F \times s_j$ random pairs of its pins and add an edge with cost one to the graph for each selected pair of pins (vertices). Note that this scheme is an experimental effort to alleviate the above mentioned problem. We use $T = 50$ and $F = 5$ in accordance to the recommendations given in [1].

## 4.2 A Multilevel Hypergraph Partitioning Heuristic

In this work, we exploit the successful multilevel methodology proposed and implemented for graph partitioning (Metis [10]) to develop a new multilevel hypergraph partitioning tool, called PaToH (PaToH: **Pa**rtitioning **To**ols for **H**ypergraphs). We should note that current implementation is just an initial implementation to experiment both the validity of our hypergraph models and the performance of multilevel approach on hypergraph partitioning.

**Coarsening Phase** In this phase, the given hypergraph $\mathcal{H} = \mathcal{H}_0$ is coarsened into a sequence of smaller hypergraphs $\mathcal{H}_1 = (\mathcal{V}_1, \mathcal{N}_1)$, $\mathcal{H}_2 = (\mathcal{V}_2, \mathcal{N}_2), \ldots, \mathcal{H}_m = (\mathcal{V}_m, \mathcal{N}_m)$ satisfying $|\mathcal{V}_0| > |\mathcal{V}_1| > |\mathcal{V}_2| > \ldots > |\mathcal{V}_m|$. This coarsening is achieved by combining vertex pairs of hypergraph $\mathcal{H}_i$ into *supernodes* of next level hypergraph $\mathcal{H}_{i+1}$. The weight of each supernode of $\mathcal{H}_{i+1}$ is set equal to the sum of its constituent vertices in $\mathcal{H}_i$. Also, the net set of each supernode is set equal to the union of the net sets of its constituent vertices. Coarsening phase terminates when number of vertices in the coarsened hypergraph reduces below 100 (i.e., $|\mathcal{V}_m| \leq 100$) following the recommendation given in [10].

In the current implementation, we use a randomized vertex matching for coarsening. In this scheme, an un-matched vertex $u$ of hypergraph $\mathcal{H}_i$ is selected randomly. Then, we consider all un-matched vertices which share nets with vertex $u$ for matching. We match $u$ with the vertex $v$ such that the sum of the costs of the shared nets between $u$ and $v$ is maximum among all considered vertices. If there exists no un-matched vertex which shares net(s) with the selected vertex $u$, then vertex $u$ is left un-matched. In rowwise matrix decomposition context (i.e., column-net model), this matching scheme corresponds to combining rows or row groups with similar row sparsity patterns. This in turn corresponds to combining rows or row groups which need similar sets of **x**-vector components in the pre-communication scheme. A dual discussion holds for columnwise matrix decomposition using the row-net model.

**Partitioning Phase** The goal in this phase is to find a partition on the coarsest hypergraph $\mathcal{H}_m$. In the current implementation we use $k$-way FM algorithm with the cutsize definition (2.b) for partitioning $\mathcal{H}_m$. Since the coarsest hypergraph $\mathcal{H}_m$ is small, we run the $k$-way FM heuristic more than once and take the minimum. We set this number of trials to 5 in PaToH.

**Uncoarsening Phase** At each level $i$ (for $i = m, m-1, \ldots, 1$), partition $\Pi_i$ found on $\mathcal{H}_i$ is projected back to the partition $\Pi_{i-1}$ on $\mathcal{H}_{i-1}$. The constituent

**Table 1.** Properties of test matrices. $d$ and $s$ denote the vertex degree and net size in graph and hypergraph models, respectively. $Z$ denotes the total number of non-zeros in matrices.

| name | $\lvert\mathcal{V}\rvert=\lvert\mathcal{N}\rvert$ | $\lvert\mathcal{E}\rvert$ | $d_{avg}$ | $Z$ | $s_{avg}$ | $s_{min}$ | $s_{max}$ |
|---|---|---|---|---|---|---|---|
| bcspwr7 | 1612 | 2106 | 1.31 | 5824 | 3.61 | 2 | 13 |
| bcspwr10 | 5300 | 8271 | 1.56 | 21842 | 4.12 | 2 | 14 |
| lshp2614 | 2614 | 7683 | 2.94 | 17980 | 6.88 | 4 | 7 |
| lshp3466 | 3466 | 10215 | 2.95 | 23896 | 6.89 | 4 | 7 |
| dwt2680 | 2680 | 11173 | 4.17 | 25026 | 9.34 | 4 | 19 |
| bcsstk21 | 3600 | 11500 | 3.19 | 26600 | 7.39 | 4 | 9 |
| ganges | 1681 | 10932 | 6.50 | 23545 | 14.01 | 3 | 86 |
| perold | 1376 | 23675 | 17.21 | 48726 | 35.41 | 3 | 86 |
| sctab2 | 1880 | 28963 | 15.41 | 59806 | 31.81 | 3 | 62 |
| sctab3 | 2480 | 38263 | 15.43 | 79006 | 31.86 | 3 | 79 |

vertices of each supernode of $\mathcal{H}_i$ is assigned to the part of their supernode. Obviously, this new partition $\Pi_{i-1}$ has the same cutsize with the previous partition $\Pi_i$. Then, we refine this partition by running our $k$-way FM algorithm on $\mathcal{H}_{i-1}$ starting from the initial partition $\Pi_{i-1}$. However, in this phase, we limit the maximum number of passes to 2. We also put an early termination rule into one pass of $k$-way FM. A pass is terminated whenever last $0.25\times\lvert\mathcal{V}_i\rvert$ moves do not decrease the cut.

## 5 Experimental Results

We have implemented $k$-way FM graph and hypergraph partitioning heuristics and a multilevel $k$-way hypergraph partitioning algorithm PaToH, and used Metis[10] graph partitioning tool for the experimental evaluation of the validity of the proposed hypergraph models. FM heuristics iteratively improve initial feasible partitions. A partition is said to be feasible if it satisfies the load balance criterion $W_{avg}(1-\varepsilon) \leq W_p \leq W_{avg}(1+\varepsilon)$, for each part $p = 1, 2, \ldots k$. Here, $W_{avg} = (\sum_{i=1}^{n} w_i)/k$ denotes the part sizes of each part under perfect load balance condition, and $\varepsilon$ represents the predetermined maximum load imbalance ratio allowed. We have used $\varepsilon = 0.04$ in all heuristics.

Symmetric sparse matrices selected from Harwell-Boeing collection [4] and linear programming problems in NETLIB suite [7] are used for experimentation. Note that test matrices are restricted to symmetric matrices since graphs cannot be used to model unsymmetric square and rectangular matrices. Table 1 displays the characteristics of the selected test matrices. BCSPWR07 and BCSPWR10 matrices come from the sparse matrix representation of power networks. LSHP2614 and LSHP3466 matrices come from the finite element discretizations of L-shaped regions. DWT2680 and BCSSTK21 are structural engineering problems. The sparsity patterns of GANGES, PEROLD, SCTAB2 and SCTAB3 are obtained from the NETLIB suite by multiplying the respective constraint matrices with their transposes. Power matrices, structural engineering matrix DWT2680, and NETLIB matrices GANGES and PEROLD have unstructured sparsity pattern. Finite element matrices, structural engineering matrix BCSSTK21, and NETLIB matrices SCTAB2–3 have rather structured sparsity pattern.

The graph and hypergraph representations of these matrices are partitioned to 4, 8, 16, and 32 parts by running the heuristics on a Sun UltraSparc 1/140. Each heuristic were run 50 times for each decomposition instance using random initial seeds. Minimum communication volume values of these 50 runs are displayed in Table 2 together with the average run-times. Communication volume values displayed in this table correspond to the communication cost computed according to (2.b).

We will refer to the FM heuristics using the graph and hypergraph models as FM-G and FM-H, respectively. As seen in Table 2, FM-H usually finds better decompositions than FM-G (8% better on the overall average). As also seen in this table, FM-H always finds drastically better decompositions than FM-G on unstructured test matrices with small net sizes (e.g., power matrices). However, the relative performance of FM-H with respect to FM-G deteriorates on structured matrices with large net sizes (e.g., finite element matrices and NETLIB matrices SCTAB2–3). This experimental finding can be attributed to the following reason. FM-H algorithm encounters large number of zero move gains during the decomposition of such matrices. FM-H algorithm randomly resolves these ties. However, on such cases, FM-G algorithm tends to gather the adjacent vertices although they do not decrease the actual communication requirement at that point in time. However, as seen in the table, PaToH overcomes the large net size problem because of the multilevel clustering (matchings) performed during the coarsening phase (e.g., PEROLD, SCTAB2–3).

In multilevel heuristics, clique-net approach does not perform very well compared to the graph model Metis (only 3% better on the overall average) as expected. As seen in Table 2, our multilevel hypergraph partitioning heuristic (PaToH) almost always performs better than the graph model Metis (13% better on the overall average). However, our current implementation of PaToH is 1.14 to 22.09 times slower than graph model Metis (6.55 times slower on the overall average). As described in Section 4.2, current implementation of the PaToH is just an initial implementation to experiment the performance of multilevel approaches on hypergraph partitioning.

## 6 Conclusion and Future Research

Two hypergraph models were proposed for decomposing sparse matrices for parallel matrix-vector multiplication. The proposed models avoid all deficiencies of the graph model. The proposed models enable the representation and hence the decomposition of unsymmetric square and rectangular matrices as well as symmetric matrices. Furthermore, they introduce a much more accurate representation for the communication requirement. The proposed models reduce the decomposition problem to the well-known *hypergraph partitioning* problem thus enabling the use of existing circuit partitioning heuristics and tools widely used in VLSI design. Fast Kernighan-Lin based graph and hypergraph partitioning

**Table 2.** Minimum communication costs for 50 runs and average run-times (in seconds). Numbers in parentheses represent values normalized with respect to the graph model results found by the same class of heuristics. Bold values indicate the best communication volume values with respective heuristics.

| name | $k$ | $k$-way FM Heuristics | | Multilevel Heuristics | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Graph Model | Hypergraph Model | Graph Model Metis | | Hypergraph Model | | | |
| | | | | | | Clique-Net Metis | | PaToH | |
| | | cost | cost | cost | time | cost | time | cost | time |
| bcspwr7 | 4 | 312 | **55** (0.18) | 33 | 0.25 | 32 (0.97) | (1.12) | **27** (0.82) | (1.14) |
| | 8 | 466 | **193** (0.41) | 87 | 0.39 | 86 (0.99) | (1.00) | **83** (0.95) | (1.74) |
| | 16 | 630 | **359** (0.57) | 176 | 0.61 | 177 (1.01) | (0.93) | **174** (0.99) | (3.73) |
| | 32 | 677 | **506** (0.75) | **304** | 1.02 | 307 (1.01) | (0.87) | 315 (1.04) | (8.37) |
| bcspwr10 | 4 | 1260 | **384** (0.30) | 124 | 0.71 | 126 (1.02) | (1.11) | **117** (0.94) | (1.73) |
| | 8 | 1935 | **727** (0.38) | 231 | 0.87 | **229** (0.99) | (1.08) | 238 (1.03) | (3.31) |
| | 16 | 2328 | **1417** (0.61) | 422 | 1.11 | **413** (0.98) | (1.07) | 414 (0.98) | (7.37) |
| | 32 | 2570 | **1739** (0.68) | 707 | 1.56 | **690** (0.98) | (1.07) | 720 (1.02) | (19.25) |
| lshp2614 | 4 | 206 | **204** (0.99) | 234 | 0.47 | 243 (1.04) | (1.31) | **207** (0.88) | (1.37) |
| | 8 | 392 | **386** (0.98) | 420 | 0.64 | 420 (1.00) | (1.25) | **396** (0.94) | (2.24) |
| | 16 | 641 | **638** (1.00) | 687 | 0.91 | 691 (1.01) | (1.14) | **662** (0.96) | (4.64) |
| | 32 | 1024 | **998** (0.97) | 1064 | 1.36 | 1069 (1.00) | (1.03) | **1048** (0.98) | (14.27) |
| lshp3466 | 4 | 235 | **234** (1.00) | 275 | 0.54 | 268 (0.97) | (1.30) | **237** (0.86) | (1.57) |
| | 8 | 449 | **447** (1.00) | 492 | 0.68 | 483 (0.98) | (1.25) | **451** (0.92) | (2.75) |
| | 16 | **728** | **728** (1.00) | 803 | 0.92 | 773 (0.96) | (1.21) | **759** (0.95) | (5.88) |
| | 32 | 1171 | **1135** (0.97) | 1219 | 1.36 | 1190 (0.98) | (1.12) | **1169** (0.96) | (16.27) |
| dwt2680 | 4 | 222 | **189** (0.85) | 205 | 0.50 | 199 (0.97) | (1.57) | **186** (0.91) | (1.90) |
| | 8 | 623 | **506** (0.81) | 497 | 0.65 | 469 (0.94) | (1.44) | **440** (0.89) | (3.01) |
| | 16 | 1119 | **970** (0.87) | 879 | 0.91 | 844 (0.96) | (1.31) | **836** (0.95) | (6.06) |
| | 32 | 1606 | **1624** (1.01) | 1438 | 1.37 | 1370 (0.95) | (1.09) | **1369** (0.95) | (17.18) |
| bcsstk21 | 4 | 244 | **240** (0.98) | 281 | 0.91 | **240** (0.85) | (1.05) | **240** (0.85) | (1.24) |
| | 8 | 1064 | **1026** (0.96) | 707 | 1.09 | 587 (0.83) | (1.07) | **554** (0.78) | (2.27) |
| | 16 | 2021 | **1859** (0.92) | 1244 | 1.37 | 1061 (0.85) | (1.06) | **993** (0.80) | (5.05) |
| | 32 | 2692 | **2405** (0.89) | 2088 | 1.88 | 1763 (0.84) | (1.02) | **1674** (0.80) | (15.33) |
| ganges | 4 | 733 | **379** (0.52) | 395 | 0.54 | 357 (0.90) | (1.19) | **297** (0.75) | (2.04) |
| | 8 | 1236 | **787** (0.64) | 713 | 0.73 | 724 (1.02) | (1.13) | **589** (0.83) | (4.06) |
| | 16 | 1771 | **1730** (0.98) | 1204 | 1.02 | 1348 (1.12) | (1.02) | **1139** (0.95) | (8.82) |
| | 32 | **2550** | 2792 (1.09) | 2167 | 1.49 | 2331 (1.08) | (0.94) | **2122** (0.98) | (22.09) |
| perold | 4 | 1320 | **1286** (0.97) | 1041 | 0.96 | 1017 (0.98) | (3.65) | **868** (0.83) | (2.29) |
| | 8 | 2283 | **2216** (0.97) | 2233 | 1.23 | 2198 (0.98) | (3.04) | **1652** (0.74) | (4.29) |
| | 16 | 4044 | **3537** (0.87) | 3750 | 1.62 | 3737 (1.00) | (2.44) | **2656** (0.71) | (7.38) |
| | 32 | 5780 | **4778** (0.83) | 5471 | 2.29 | 5170 (0.94) | (1.89) | **4111** (0.75) | (14.58) |
| sctab2 | 4 | **1095** | 1840 (1.68) | 1106 | 1.31 | 1060 (0.96) | (4.17) | **901** (0.81) | (1.78) |
| | 8 | **1930** | 2882 (1.49) | 1976 | 1.78 | 1859 (0.94) | (3.28) | **1522** (0.77) | (2.80) |
| | 16 | **3672** | 4213 (1.15) | 3245 | 2.33 | 3058 (0.94) | (2.66) | **2563** (0.79) | (4.94) |
| | 32 | 6653 | **6019** (0.90) | 5624 | 3.08 | 5414 (0.96) | (2.18) | **4151** (0.74) | (11.69) |
| sctab3 | 4 | **1397** | 2511 (1.80) | 1396 | 1.61 | 1281 (0.92) | (3.49) | **1119** (0.80) | (1.99) |
| | 8 | **2642** | 3788 (1.43) | 2534 | 2.02 | 2315 (0.91) | (2.97) | **1943** (0.77) | (3.13) |
| | 16 | **4484** | 5439 (1.21) | 4007 | 2.54 | 3907 (0.98) | (2.52) | **3163** (0.79) | (5.87) |
| | 32 | **7531** | 7619 (1.01) | 6802 | 3.27 | 6417 (0.94) | (2.12) | **4944** (0.73) | (16.46) |

heuristics were implemented and the successful multilevel graph partitioning tool (Metis) was used for the experimental evaluation of the validity of the proposed hypergraph models. An initial version for a multilevel hypergraph partitioning heuristic was also implemented for experimenting both the validity of the proposed models and the performance of the multilevel approach on hypergraph partitioning. Experimental results on sparse matrices, selected from Harwell-Boeing collection and NETLIB suite, confirmed the validity of our proposed hypergraph models. Initial experimental results were also found to be promising for the performance of multilevel approaches on hypergraph partitioning. We are currently working on improving both the speed and quality performance of our multilevel hypergraph partitioning heuristic.

## References

1. C. J. Alpert, L. W. Hagen, and A. B. Kahng. A hybrid multilevel/genetic approach for circuit partitioning. Technical report, UCLA CS Dept., 1996.
2. C. J. Alpert and A. B. Kahng. Recent directions in netlist partitioning: A survey. *VLSI Journal*, 19(1-2):1–81, 1995.
3. Ü. V. Çatalyürek and C. Aykanat. A hypergraph model for mapping repeated sparse matrix-vector product computations onto multicomputers. In *Proceedings of International Conference on Hyperformance Computing*, December 1995.
4. I. S. Duff, R. Grimes, and J. Lewis. Sparse matrix test problems. *ACM Transactions on Mathematical Software*, 15(1):1–14, march 1989.
5. F. Erçal, C. Aykanat, F. Özgüner and P. Sadayappan. Iterative algorithms for solution of large sparse systems of linear equations on hypercubes. *IEEE Transactions on Computers*, 37:1554–1567, 1988.
6. C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *Proceedings of the 19th ACM/IEEE Design Automation Conference*, pages 175–181, 1982.
7. D. M. Gay. Electronic mail distribution of linear programming test problems. *Mathematical Programming Society COAL Newsletter*, 1985.
8. B. Hendrickson and R. Leland. A multilevel algorithm for partitioning graphs. Technical report, Sandia National Laboratories, 1993.
9. B. A. Hendrickson, W. Camp, S. J. Plimpton and R. W. Leland. Massively parallel methods for engineering and science problems. *Communication of ACM*, 37(4):31–41, April 1994.
10. G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. Tech. report, CS Dept., University of Minnesota, 1995.
11. B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307, February 1970.
12. C. Pommerell, M. Annaratone, and W. Fichtner. A set of new mapping and coloring heuristics for distributed-memory parallel processors. *SIAM Journal of Scientific and Statistical Computing*, 13(1):194–226, January 1992.
13. J. Ramanujam, F. Erçal and P. Sadayappan. Task allocation onto a hypercube by recursive mincut bipartitioning. *J. Parallel and Dist. Comput.*, 10:35–44, 1990.
14. D. G. Schweikert and B. W. Kernighan. A proper model for the partitioning of electrical circuits. In *Proceedings of the 9th ACM/IEEE Design Automation Conference*, pages 57–62, 1972.

This article was processed using the LaTeX macro package with LLNCS style