Hypergraph-Partitioning Based Decomposition for Parallel Sparse-Matrix Vector Multiplication[*]

Ümit V. Çatalyürek and Cevdet Aykanat, Member, IEEE
Computer Engineering Department, Bilkent University
06533 Bilkent, Ankara, Turkey
{cumit/aykanat}@cs.bilkent.edu.tr

Abstract

In this work, we show that the standard graph-partitioning based decomposition of sparse matrices does not reflect the actual communication volume requirement for parallel matrix-vector multiplication. We propose two computational hypergraph models which avoid this crucial deficiency of the graph model. The proposed models reduce the decomposition problem to the well-known hypergraph partitioning problem. The recently proposed successful multilevel framework is exploited to develop a multilevel hypergraph partitioning tool PaToH for the experimental verification of our proposed hypergraph models. Experimental results on a wide range of realistic sparse test matrices confirm the validity of the proposed hypergraph models. In the decomposition of the test matrices, the hypergraph models using PaToH and hMeTiS result in up to 63% less communication volume (30%–38% less on the average) than the graph model using MeTiS, while PaToH is only 1.3–2.3 times slower than MeTiS on the average.

Index Terms—Sparse matrices, matrix multiplication, parallel processing, matrix decomposition, computational graph model, graph partitioning, computational hypergraph model, hypergraph partitioning.

[*] 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.

1 INTRODUCTION

Iterative solvers are widely used for the solution of large, sparse, linear system of equations on multicomputers. Two basic types of operations are repeatedly performed at each iteration. These are linear operations on dense vectors and sparse-matrix vector product (SpMxV) of the form **y**=**Ax**, where **A** is an $m \times m$ square matrix with the same sparsity structure as the coefficient matrix [3, 5, 8, 35], and **y** and **x** are dense vectors. Our goal is the parallelization of the computations in the iterative solvers through *rowwise* or *columnwise* decomposition of the **A** matrix as

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1^r \\ \vdots \\ \mathbf{A}_k^r \\ \vdots \\ \mathbf{A}_K^r \end{bmatrix} \qquad and \qquad \mathbf{A} = \left[\mathbf{A}_1^c \cdots \mathbf{A}_k^c \cdots \mathbf{A}_K^c\right],$$

where processor P_k owns row stripe $\mathbf{A}_k^r$ or column stripe $\mathbf{A}_k^c$, respectively, for a parallel system with K processors. In order to avoid the communication of vector components during the linear vector operations, a symmetric partitioning scheme is adopted. That is, all vectors used in the solver are divided conformally with the row partitioning or the column partitioning in rowwise or columnwise decomposition schemes, respectively. In particular, the **x** and **y** vectors are divided as $[\mathbf{x}_1, \ldots, \mathbf{x}_K]^t$ and $[\mathbf{y}_1, \ldots, \mathbf{y}_K]^t$, respectively. In rowwise decomposition, processor P_k is responsible for computing $\mathbf{y}_k = \mathbf{A}_k^r\mathbf{x}$ and the linear operations on the k-th blocks of the vectors. In columnwise decomposition, processor P_k is responsible for computing $\mathbf{y}^k = \mathbf{A}_k^c\mathbf{x}_k$ (where $\mathbf{y} = \sum_{k=1}^K \mathbf{y}^k$) and the linear operations on the k-th blocks of the vectors. With these decomposition schemes, the linear vector operations can be easily and efficiently parallelized [3, 35], such that only the inner-product computations introduce global communication overhead of which its volume does not scale up with increasing problem size. In parallel SpMxV, the rowwise and columnwise decomposition schemes require communication before or after the local SpMxV computations, thus they can also be considered as *pre* and *post* communication schemes, respectively. Depending on the way in which the rows or columns of **A** are partitioned among the processors, entries in **x** or entries in $\mathbf{y}^k$ may need to be communicated among the processors. Unfortunately, the communication volume scales up with increasing problem size. Our goal is to find a rowwise or columnwise partition of **A** that minimizes the total volume of communication while maintaining the computational load balance.

The decomposition heuristics [32, 33, 37] proposed for computational load balancing may result in extensive communication volume, because they do not consider the minimization of the communication volume during the decomposition. In one-dimensional (1D) decomposition, the worst-case communication requirement is $K(K-1)$ messages and $(K-1)m$ words, and it occurs when each submatrix $\mathbf{A}_k^r$ ($\mathbf{A}_k^c$) has at least one nonzero in each column (row) in rowwise (columnwise) decomposition. The approach based on 2D checkerboard partitioning [15, 30] reduces the worst-case communication to $2K(\sqrt{K}-1)$ messages and $2(\sqrt{K}-1)m$ words. In this approach, the worst-case occurs when each row and column of each submatrix has at least one nonzero.

The computational graph model is widely used in the representation of computational structures of various scientific applications, including repeated SpMxV computations, to decompose the computational domains for parallelization [5, 6, 20, 21, 27, 28, 31, 36]. In this model, the problem of sparse matrix decomposition for minimizing the communication volume while maintaining the load balance is formulated as the well-known K-way graph partitioning problem. In this work, we show the deficiencies of the graph model for decomposing sparse matrices for parallel SpMxV. The first deficiency is that it can only be used for structurally symmetric square matrices. In order to avoid this deficiency, we propose a generalized graph model in Section 2.3 which enables the decomposition of structurally nonsymmetric square matrices as well as symmetric matrices. The second deficiency is the fact that the graph models (both standard and proposed ones) do not reflect the actual communication requirement as will be described in Section 2.4. These flaws are also mentioned in a concurrent work [16]. In this work, we propose two *computational hypergraph* models which avoid all deficiencies of the graph model. The proposed models enable the representation and hence the decomposition of rectangular matrices [34] as well as symmetric and nonsymmetric square matrices. Furthermore, they introduce an exact representation for the communication volume requirement as described in Section 3.2. The proposed hypergraph models reduce the decomposition problem to the well-known *K-way 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 in VLSI community.

Decomposition is a preprocessing introduced for the sake of efficient parallelization of a given problem. Hence, heuristics used for decomposition should run in low order polynomial time. Recently, multilevel graph partitioning heuristics [4, 13, 21] are proposed leading to fast and successful graph partitioning tools Chaco [14] and MeTiS [22]. We have exploited the multilevel partitioning methods for the experimental verification of the 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 randomized clique-net model as presented in Section 4.1. In the second approach, the lack of a multilevel hypergraph partitioning tool at the time of this work was carried led us to develop a multilevel hypergraph partitioning tool PaToH for a fair experimental comparison of the hypergraph models with the graph models. Another objective in our PaToH implementation was to investigate the performance of multilevel approach in hypergraph partitioning as described in Section 4.2. Recently released multilevel hypergraph partitioning tool hMeTiS [24] is also used in the second approach. Experimental results presented in Section 5 confirm both the validity of our proposed hypergraph models and the appropriateness of the multilevel approach to hypergraph partitioning. The hypergraph models using PaToH and hMeTiS produce 30%–38% better decompositions than the graph models using MeTiS, while the hypergraph models using PaToH are only 34%–130% slower than the graph models using the most recent version (Version 3.0) of MeTiS, on the average.

2 GRAPH MODELS AND THEIR DEFICIENCIES

2.1 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 distinct 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. Weights and costs can be assigned to the vertices and edges of the graph, respectively. 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.

$\Pi=\{\mathcal{P}_1,\mathcal{P}_2,\ldots,\mathcal{P}_K\}$ is a *K-way partition* of $\mathcal{G}$ if the following conditions hold: each part $\mathcal{P}_k, 1\leq k\leq K$, is a nonempty subset of $\mathcal{V}$, parts are pairwise disjoint ($\mathcal{P}_k\cap\mathcal{P}_\ell=\emptyset$ for all $1\leq k<\ell\leq K$), and union of K parts is equal to $\mathcal{V}$ (i.e. $\bigcup_{k=1}^{K}\mathcal{P}_k=\mathcal{V}$). A K-way partition is also called a *multiway* partition if $K>2$ and a *bipartition* if $K=2$. A partition is said to be balanced if each part $\mathcal{P}_k$ satisfies the *balance criterion*

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

In (1), weight W_k of a part $\mathcal{P}_k$ is defined as the sum of the weights of the vertices in that part (i.e. $W_k=\sum_{v_i\in\mathcal{P}_k}w_i$), $W_{avg}=(\sum_{v_i\in\mathcal{V}}w_i)/K$ 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 $\mathcal{G}$, an edge is said to be *cut* if its pair of vertices belong to two different parts, and *uncut* otherwise. The cut and uncut edges are also referred to here as *external* and *internal* edges, respectively. The set of external edges of a partition Π is denoted as $\mathcal{E}_E$. The *cutsize* definition for representing the cost $\chi(\Pi)$ of a partition Π is

$$\chi(\Pi)=\sum_{e_{ij}\in\mathcal{E}_E}c_{ij}. \tag{2}$$

In (2), each cut edge e_{ij} contributes its cost c_{ij} to the cutsize. Hence, the 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 the balance criterion (1) on part weights is maintained. The graph partitioning problem is known to be NP-hard even for bipartitioning unweighted graphs [11].

2.2 Standard Graph Model for Structurally Symmetric Matrices

A structurally symmetric sparse matrix **A** can be represented as an undirected graph $\mathcal{G}_A=(\mathcal{V},\mathcal{E})$, where the sparsity pattern of **A** corresponds to the adjacency matrix representation of graph $\mathcal{G}_A$. That is, the vertices of $\mathcal{G}_A$ correspond to the rows/columns of matrix **A**, and there exist an edge $e_{ij}\in\mathcal{E}$ for $i\neq j$ if and only if off-diagonal entries a_{ij} and a_{ji} of matrix **A** are nonzeros. In rowwise decomposition, each vertex $v_i\in\mathcal{V}$ corresponds to atomic task i of computing the inner product of row i with column vector **x**. In columnwise decomposition, each vertex $v_i\in\mathcal{V}$ corresponds to atomic task i of computing the sparse SAXPY/DAXPY operation $\mathbf{y}=\mathbf{y}+x_i\mathbf{a}_{*i}$, where $\mathbf{a}_{*i}$ denotes column i of matrix **A**. Hence, each nonzero entry in a row and column of **A** incurs a multiply-and-add operation during the local SpMxV computations in the pre and post communication schemes, respectively. Thus, computational load w_i of row/column i is the number of nonzero entries in row/column i. In graph theoretical notation, $w_i = d_i$ when $a_{ii} = 0$ and $w_i = d_i + 1$ when $a_{ii} \neq 0$. Note that the number of nonzeros in row i and column i are equal in a symmetric matrix.

This graph model displays a bidirectional computational interdependency view for SpMxV. Each edge $e_{ij} \in \mathcal{E}$ can be considered as incurring the computations $y_i \leftarrow y_i + a_{ij}x_j$ and $y_j \leftarrow y_j + a_{ji}x_i$. Hence, each edge represents the bidirectional interaction between the respective pair of vertices in both inner and outer product computation schemes for SpMxV. If rows (columns) i and j are assigned to the same processor in a rowwise (columnwise) decomposition, then edge e_{ij} does not incur any communication. However, in the pre-communication scheme, if rows i and j are assigned to different processors then cut edge e_{ij} necessitates the communication of two floating–point words because of the need of the exchange of updated x_i and x_j values between atomic tasks i and j just before the local SpMxV computations. In the post-communication scheme, if columns i and j are assigned to different processors then cut edge e_{ij} necessitates the communication of two floating–point words because of the need of the exchange of partial y_i and y_j values between atomic tasks i and j just after the local SpMxV computations. Hence, by setting $c_{ij} = 2$ for each edge $e_{ij} \in \mathcal{E}$, both rowwise and columnwise decompositions of matrix **A** reduce to the K-way partitioning of its associated graph $\mathcal{G}_A$ according to the cutsize definition given in (2). Thus, minimizing the cutsize is an effort towards minimizing the total volume of interprocessor communication. Maintaining the balance criterion (1) corresponds to maintaining the computational load balance during local SpMxV computations.

Each vertex $v_i \in \mathcal{V}$ effectively represents both row i and column i in $\mathcal{G}_A$ although its atomic task definition differs in rowwise and columnwise decompositions. Hence, a partition Π of $\mathcal{G}_A$ automatically achieves a symmetric partitioning by inducing the same partition on the **y**-vector and **x**-vector components since a vertex $v_i \in \mathcal{P}_k$ corresponds to assigning row i (column i), y_i and x_i to the same part in rowwise (columnwise) decomposition.

In matrix theoretical view, the symmetric partitioning induced by a partition Π of $\mathcal{G}_A$ can also be considered as inducing a partial symmetric permutation on the rows and columns of **A**. Here, the partial permutation corresponds to ordering the rows/columns assigned to part P_k before the rows/columns assigned to part P_{k+1}, for $k = 1, \ldots, K-1$, where the rows/columns within a part are ordered arbitrarily. Let $\mathbf{A}^{\Pi}$ denote the permuted version of **A** according to a partial symmetric permutation induced by Π. An internal edge e_{ij} of a part $\mathcal{P}_k$ corresponds to locating both a_{ij} and a_{ji} in diagonal block $\mathbf{A}^{\Pi}_{kk}$. An external edge e_{ij} of cost 2 between parts $\mathcal{P}_k$ and $\mathcal{P}_\ell$ corresponds to locating nonzero entry a_{ij} of **A** in off-diagonal block $\mathbf{A}^{\Pi}_{k\ell}$ and a_{ji} of **A** in off-diagonal block $\mathbf{A}^{\Pi}_{\ell k}$, or vice versa. Hence, minimizing the cutsize in the graph model can also be considered as permuting the rows and columns of the matrix to minimize the total number of nonzeros in the off-diagonal blocks.

Figure 1 illustrates a sample 10×10 symmetric sparse matrix **A** and its associated graph $\mathcal{G}_A$. The numbers inside the circles indicate the computational weights of the respective vertices (rows/columns). This figure also illustrates a rowwise decomposition of the symmetric **A** matrix and the corresponding bipartitioning of $\mathcal{G}_A$ for a two–processor system. As seen in Fig. 1, the cutsize in the given graph bipartitioning is 8 which is also equal to the total number of nonzero entries in the off-diagonal blocks. The bipartition illustrated in Fig. 1 achieves perfect

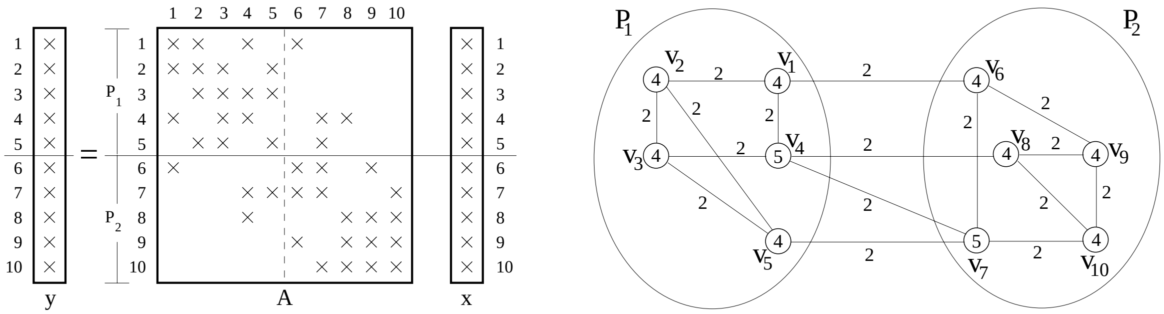


Figure 1: Two-way rowwise decomposition of a sample structurally symmetric matrix **A** and the corresponding bipartitioning of its associated graph $\mathcal{G}_A$.

load balance by assigning 21 nonzero entries to each row stripe. This number can also be obtained by adding the weights of the vertices in each part.

2.3 Generalized Graph Model for Structurally Symmetric/Nonsymmetric Square Matrices

The standard graph model is not suitable for the partitioning of nonsymmetric matrices. A recently proposed *bipartite* graph model [17, 26] enables the partitioning of rectangular as well as structurally symmetric/nonsymmetric square matrices. In this model, each row and column is represented by a vertex, and the sets of vertices representing the rows and columns form the bipartition, i.e. $\mathcal{V} = \mathcal{V}_{\mathcal{R}} \cup \mathcal{V}_{\mathcal{C}}$. There exists an edge between a row vertex $i \in \mathcal{V}_{\mathcal{R}}$ and a column vertex $j \in \mathcal{V}_{\mathcal{C}}$ if and only if the respective entry a_{ij} of matrix **A** is nonzero. Partitions $\Pi_{\mathcal{R}}$ and $\Pi_{\mathcal{C}}$ on $\mathcal{V}_{\mathcal{R}}$ and $\mathcal{V}_{\mathcal{C}}$, respectively, determine the overall partition $\Pi = \{\mathcal{P}_1, \ldots, \mathcal{P}_K\}$, where $\mathcal{P}_k = \mathcal{V}_{\mathcal{R}_k} \cup \mathcal{V}_{\mathcal{C}_k}$ for $k = 1, \ldots, K$. For rowwise (columnwise) decomposition, vertices in $\mathcal{V}_{\mathcal{R}}$ ($\mathcal{V}_{\mathcal{C}}$) are weighted with the number of nonzeros in the respective row (column) so that the balance criterion (1) is imposed only on the partitioning of $\mathcal{V}_{\mathcal{R}}$ ($\mathcal{V}_{\mathcal{C}}$). As in the standard graph model, minimizing the number of cut edges corresponds to minimizing the total number of nonzeros in the off-diagonal blocks. This approach has the flexibility of achieving nonsymmetric partitioning. In the context of parallel SpMxV, the need for symmetric partitioning on square matrices is achieved by enforcing $\Pi_{\mathcal{R}} \equiv \Pi_{\mathcal{C}}$. Hendrickson and Kolda [17] propose several bipartite-graph partitioning algorithms that are adopted from the techniques for the standard graph model and one partitioning algorithm that is specific to bipartite graphs.

In this work, we propose a simple yet effective graph model for symmetric partitioning of structurally nonsymmetric square matrices. The proposed model enables the use of the standard graph partitioning tools without any modification. In the proposed model, a nonsymmetric square matrix **A** is represented as an undirected graph $\mathcal{G}_{\mathcal{R}} = (\mathcal{V}_{\mathcal{R}}, \mathcal{E})$ and $\mathcal{G}_{\mathcal{C}} = (\mathcal{V}_{\mathcal{C}}, \mathcal{E})$ for the rowwise and columnwise decomposition schemes, respectively. Graphs $\mathcal{G}_{\mathcal{R}}$ and $\mathcal{G}_{\mathcal{C}}$ differ only in their vertex weight definitions. The vertex set and the corresponding atomic task definitions are identical to those of the symmetric matrices. That is, weight w_i of a vertex $v_i \in \mathcal{V}_{\mathcal{R}}$ ($v_i \in \mathcal{V}_{\mathcal{C}}$) is equal to the total number of nonzeros in row i (column i) in $\mathcal{G}_{\mathcal{R}}$ ($\mathcal{G}_{\mathcal{C}}$). In the edge set $\mathcal{E}$, $e_{ij} \in \mathcal{E}$ if and only if off-diagonal entries $a_{ij} \neq 0$ *or* $a_{ji} \neq 0$. That is, the vertices in the adjacency list of a vertex v_i denote the union of the column indices of the off-diagonal nonzeros at row i and the row indices of the off-diagonal nonzeros at column i. The cost c_{ij} of an edge e_{ij} is set to 1 if either $a_{ij} \neq 0$ or $a_{ji} \neq 0$, and it is set to 2 if both $a_{ij} \neq 0$ and $a_{ji} \neq 0$. The

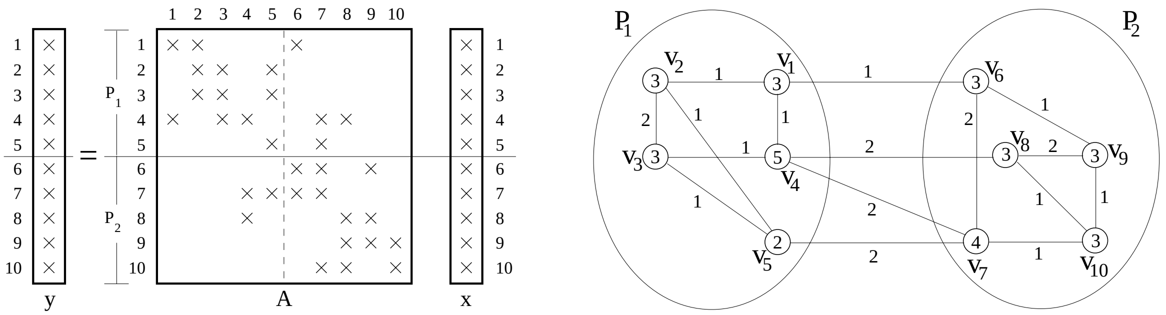


Figure 2: Two-way rowwise decomposition of a sample structurally nonsymmetric matrix **A** and the corresponding bipartitioning of its associated graph $\mathcal{G}_{\mathcal{R}}$.

proposed scheme is referred to here as a generalized model since it automatically produces the standard graph representation for structurally symmetric matrices by computing the same cost of 2 for every edge.

Figure 2 illustrates a sample 10×10 nonsymmetric sparse matrix **A** and its associated graph $\mathcal{G}_{\mathcal{R}}$ for rowwise decomposition. The numbers inside the circles indicate the computational weights of the respective vertices (rows). This figure also illustrates a rowwise decomposition of the matrix and the corresponding bipartitioning of its associated graph for a two–processor system. As seen in Fig. 2, the cutsize of the given graph bipartitioning is 7 which is also equal to the total number of nonzero entries in the off-diagonal blocks. Hence, similar to the standard and bipartite graph models, minimizing cutsize in the proposed graph model corresponds to minimizing the total number of nonzeros in the off-diagonal blocks. As seen in Fig. 2, the bipartitioning achieves perfect load balance by assigning 16 nonzero entries to each row stripe. As mentioned earlier, the $\mathcal{G}_{\mathcal{C}}$ model of a matrix for columnwise decomposition differs from the $\mathcal{G}_{\mathcal{R}}$ model only in vertex weights. Hence, the graph bipartitioning illustrated in Fig. 2 can also be considered as incurring a slightly imbalanced (15 versus 17 nonzeros) columnwise decomposition of sample matrix **A** (shown by vertical dash line) with identical communication requirement.

2.4 Deficiencies of the Graph Models

Consider the symmetric matrix decomposition given in Fig. 1. Assume that parts $\mathcal{P}_1$ and $\mathcal{P}_2$ are mapped to processors P_1 and P_2, respectively. The cutsize of the bipartition shown in this figure is equal to $2\times4=8$, thus estimating the communication volume requirement as 8 words. In the pre-communication scheme, off-block-diagonal entries $a_{4,7}$ and $a_{5,7}$ assigned to processor P_1 display the same need for the nonlocal **x**-vector component x_7 twice. However, it is clear that processor P_2 will send x_7 only once to processor P_1. Similarly, processor P_1 will send x_4 only once to processor P_2 because of the off-block-diagonal entries $a_{7,4}$ and $a_{8,4}$ assigned to processor P_2. In the post-communication scheme, the graph model treats the off-block-diagonal nonzeros $a_{7,4}$ and $a_{7,5}$ in $\mathcal{P}_1$ as if processor P_1 will send two multiplication results $a_{7,4}x_4$ and $a_{7,5}x_5$ to processor P_2. However, it is obvious that processor P_1 will compute the partial result for the nonlocal **y**-vector component $y'_7 = a_{7,4}x_4 + a_{7,5}x_5$ during the local SpMxV phase and send this single value to processor P_2 during the post-communication phase. Similarly, processor P_2 will only compute and send the single value $y'_4 = a_{4,7}x_7 + a_{4,8}x_8$ to processor P_1. Hence, the actual communication volume is in fact 6 words instead of 8 in both pre and post communication schemes. A similar analysis of the rowwise decomposition of the nonsymmetric matrix given in Fig. 2 reveals the fact that the actual communication requirement is 5 words (x_4, x_5, x_6, x_7 and x_8) instead of 7 determined by the cutsize of the given bipartition of $\mathcal{G}_\mathcal{R}$.

In matrix theoretical view, the nonzero entries in the same column of an off-diagonal block incur the communication of a single x value in the rowwise decomposition (pre-communication) scheme. Similarly, the nonzero entries in the same row of an off-diagonal block incur the communication of a single y value in the columnwise decomposition (post-communication) scheme. However, as mentioned earlier, the graph models try to minimize the total number of off-block-diagonal nonzeros without considering the relative spatial locations of such nonzeros. In other words, the graph models treat all off-block-diagonal nonzeros in an identical manner by assuming that each off-block-diagonal nonzero will incur a distinct communication of a single word.

In graph theoretical view, the graph models treat all cut edges of equal cost in an identical manner while computing the cutsize. However, r cut edges, each of cost 2, stemming from a vertex v_{i_1} in part $\mathcal{P}_k$ to r vertices $v_{i_2}, v_{i_3}, \ldots, v_{i_{r+1}}$ in part $\mathcal{P}_\ell$ incur only $r+1$ communications instead of $2r$ in both pre and post communication schemes. In the pre-communication scheme, processor P_k sends x_{i_1} to processor P_ℓ while P_ℓ sends $x_{i_2}, x_{i_3}, \ldots, x_{i_{r+1}}$ to P_k. In the post-communication scheme, processor P_ℓ sends $y'_{i_2}, y'_{i_3}, \ldots, y'_{i_{r+1}}$ to processor P_k while P_k sends y'_{i_1} to P_ℓ. Similarly, the amount of communication required by r cut edges, each of cost 1, stemming from a vertex v_{i_1} in part $\mathcal{P}_k$ to r vertices $v_{i_2}, v_{i_3}, \ldots, v_{i_{r+1}}$ in part $\mathcal{P}_\ell$ may vary between 1 and r words instead of exactly r words determined by the cutsize of the given graph partitioning.

3 HYPERGRAPH MODELS FOR DECOMPOSITION

3.1 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., $n_j \subseteq \mathcal{V}$. The 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., $s_j = |pins[n_j]|$. 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., $d_i = |nets[v_i]|$. Graph is a special instance of hypergraph such that each net has exactly two pins. Similar to graphs, 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 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* if it connects more than one part (i.e. $\lambda_j > 1$), and *uncut* otherwise (i.e. $\lambda_j = 1$). The cut and uncut nets are also referred to here as *external* and *internal* nets, respectively. The set of external nets of a partition Π is denoted as $\mathcal{N}_E$. There are various *cutsize* definitions for representing the cost $\chi(\Pi)$ of a partition Π. Two relevant definitions are:

$$(a) \quad \chi(\Pi) = \sum_{n_j \in \mathcal{N}_E} c_j \qquad and \qquad (b) \quad \chi(\Pi) = \sum_{n_j \in \mathcal{N}_E} c_j(\lambda_j - 1). \tag{3}$$

In (3.a), the cutsize is equal to the sum of the costs of the cut nets. In (3.b), each cut net n_j contributes $c_j(\lambda_j - 1)$

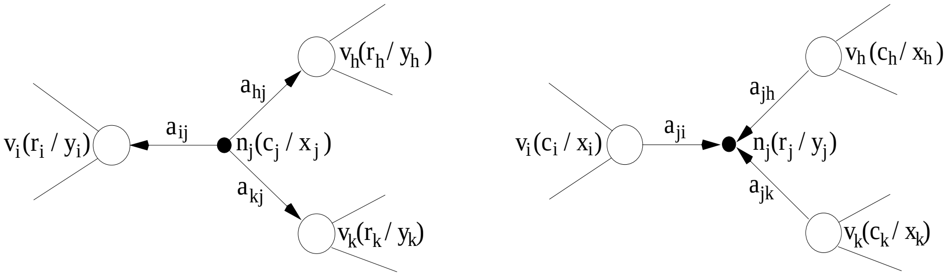
to the cutsize. Hence, the hypergraph partitioning problem [29] can be defined as the task of dividing a hypergraph into two or more parts such that the cutsize is minimized, while a given balance criterion (1) among the part weights is maintained. Here, part weight definition is identical to that of the graph model. The hypergraph partitioning problem is known to be NP-hard [29].

3.2 Two Hypergraph Models for Decomposition

We propose two computational hypergraph models for the decomposition of sparse matrices. These models are referred to here as the *column-net* and *row-net* models proposed for the rowwise decomposition (pre-communication) and columnwise decomposition (post-communication) schemes, respectively.

In the column-net model, matrix **A** is represented as a hypergraph $\mathcal{H}_\mathcal{R}=(\mathcal{V}_\mathcal{R},\mathcal{N}_\mathcal{C})$ for rowwise decomposition. Vertex and net sets $\mathcal{V}_\mathcal{R}$ and $\mathcal{N}_\mathcal{C}$ correspond to the rows and columns of matrix **A**, respectively. There exist one vertex v_i and one net n_j for each row i and column j, respectively. Net $n_j \subseteq \mathcal{V}_\mathcal{R}$ contains the vertices corresponding to the rows which have a nonzero entry in 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 atomic task i of computing the inner product of row i with column vector **x**. Hence, computational weight w_i of a vertex $v_i \in \mathcal{V}_\mathcal{R}$ is equal to the total number of nonzeros in row i. The nets of $\mathcal{H}_\mathcal{R}$ represent the *dependency* relations of the atomic tasks on the **x**-vector components in rowwise decomposition. Each net n_j can be considered as incurring the computation $y_i \leftarrow y_i + a_{ij}x_j$ for each vertex (row) $v_i \in n_j$. Hence, each net n_j denotes the set of atomic tasks (vertices) that need x_j. Note that each pin v_i of a net n_j corresponds to a unique nonzero a_{ij} thus enabling the representation and decomposition of structurally nonsymmetric matrices as well as symmetric matrices without any extra effort. Figure 3(a) illustrates the dependency relation view of the column-net model. As seen in this figure, net $n_j=\{v_h, v_i, v_k\}$ represents the dependency of atomic tasks h, i, k to x_j because of the computations $y_h \leftarrow y_h + a_{hj}x_j$, $y_i \leftarrow y_i + a_{ij}x_j$ and $y_k \leftarrow y_k + a_{kj}x_j$. Figure 4(b) illustrates the column-net representation of the sample 16×16 nonsymmetric matrix given in Fig. 4(a). In Fig. 4(b), the pins of net $n_7=\{v_7, v_{10}, v_{13}\}$ represent nonzeros $a_{7,7}$, $a_{10,7}$, and $a_{13,7}$. Net n_7 also represents the dependency of atomic tasks 7, 10 and 13 to x_7 because of the computations $y_7 \leftarrow y_7 + a_{7,7}x_7$, $y_{10} \leftarrow y_{10} + a_{10,7}x_7$ and $y_{13} \leftarrow y_{13} + a_{13,7}x_7$.

The row-net model can be considered as the dual of the column-net model. In this model, matrix **A** is represented as a hypergraph $\mathcal{H}_\mathcal{C}=(\mathcal{V}_\mathcal{C},\mathcal{N}_\mathcal{R})$ for columnwise decomposition. Vertex and net sets $\mathcal{V}_\mathcal{C}$ and $\mathcal{N}_\mathcal{R}$ correspond to the columns and rows of matrix **A**, respectively. There exist one vertex v_i and one net n_j for each column i and row j, respectively. Net $n_j \subseteq \mathcal{V}_\mathcal{C}$ contains the vertices corresponding to the columns which have a nonzero entry in 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 atomic task i of computing the sparse SAXPY/DAXPY operation $\mathbf{y}=\mathbf{y}+x_i\mathbf{a}_{*i}$. Hence, computational weight w_i of a vertex $v_i \in \mathcal{V}_\mathcal{C}$ is equal to the total number of nonzeros in column i. The nets of $\mathcal{H}_\mathcal{C}$ represent the *dependency* relations of the computations of the **y**-vector components on the atomic tasks represented by the vertices of $\mathcal{H}_\mathcal{C}$ in columnwise decomposition. Each net n_j can be considered as incurring the computation $y_j \leftarrow y_j + a_{ji}x_i$ for each vertex (column) $v_i \in n_j$. Hence, each net n_j denotes the set of atomic task results needed to accumulate y_j. Note that each pin v_i of a net n_j corresponds to a unique nonzero a_{ji} thus enabling the representation and decomposition of



(a) Column-Net Model

(b) Row-Net Model

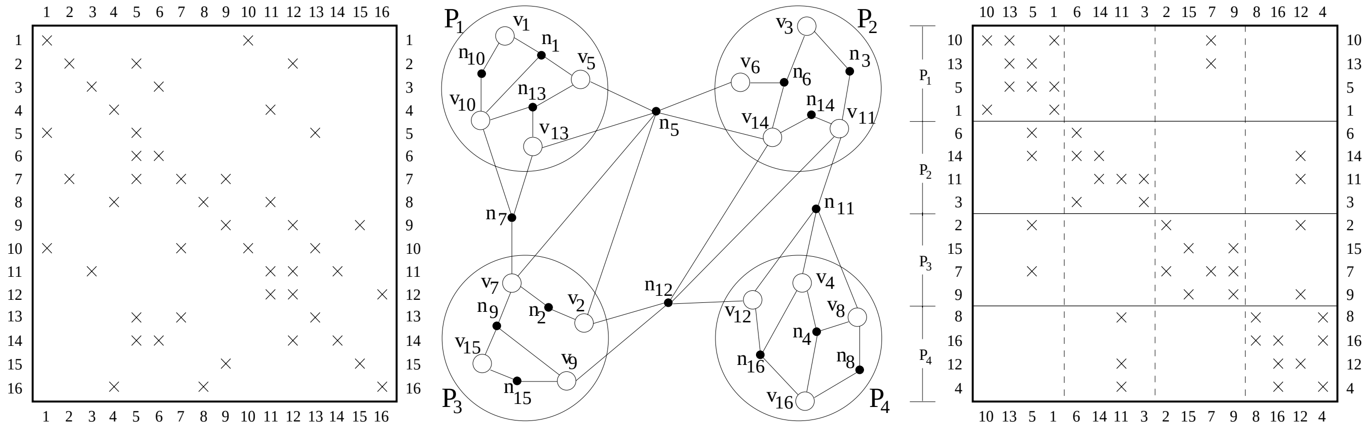
Figure 3: Dependency relation views of (a) column-net and (b) row-net models.

structurally nonsymmetric matrices as well as symmetric matrices without any extra effort. Figure 3(b) illustrates the dependency relation view of the row-net model. As seen in this figure, net $n_j = \{v_h, v_i, v_k\}$ represents the dependency of accumulating $y_j = y_j^h + y_j^i + y_j^k$ on the partial y_j results $y_j^h = a_{jh}x_h, y_j^i = a_{ji}x_i$ and $y_j^k = a_{jk}x_k$. Note that the row-net and column-net models become identical in structurally symmetric matrices.

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 to the K-way hypergraph partitioning problem according to the cutsize definition given in (3.b) for the pre and post communication schemes, respectively. Consistency of the proposed hypergraph models for accurate representation of communication volume requirement while maintaining the symmetric partitioning restriction depends on the condition that "$v_j \in n_j$ for each net n_j". We first assume that this condition holds in the discussion throughout the following four paragraphs and then discuss the appropriateness of the assumption in the last paragraph of this section.

The validity of the proposed hypergraph models is discussed only for the column-net model. A dual discussion holds for the row-net model. Consider a partition Π of $\mathcal{H}_\mathcal{R}$ in the column-net model for rowwise decomposition of a matrix **A**. Without loss of generality, we assume that part $\mathcal{P}_k$ is assigned to processor P_k for $k = 1, 2, \ldots, K$. As Π is defined as a partition on the vertex set of $\mathcal{H}_\mathcal{R}$, it induces a complete part (hence processor) assignment for the rows of matrix **A** and hence for the components of the **y** vector. That is, a vertex v_i assigned to part $\mathcal{P}_k$ in Π corresponds to assigning row i and y_i to part $\mathcal{P}_k$. However, partition Π does not induce any part assignment for the nets of $\mathcal{H}_\mathcal{R}$. Here, we consider partition Π as inducing an assignment for the internal nets of $\mathcal{H}_\mathcal{R}$ hence for the respective **x**-vector components. Consider an internal net n_j of part $\mathcal{P}_k$ (i.e. $\Lambda_j = \{\mathcal{P}_k\}$) which corresponds to column j of **A**. As all pins of net n_j lie in $\mathcal{P}_k$, all rows (including row j by the consistency condition) which need x_j for inner-product computations are already assigned to processor P_k. Hence, internal net n_j of $\mathcal{P}_k$, which does not contribute to the cutsize (3.b) of partition Π, does not necessitate any communication if x_j is assigned to processor P_k. The assignment of x_j to processor P_k can be considered as permuting column j to part $\mathcal{P}_k$, thus respecting the symmetric partitioning of **A** since row j is already assigned to $\mathcal{P}_k$. In the 4-way decomposition given in Fig. 4(b), internal nets n_1, n_{10}, n_{13} of part $\mathcal{P}_1$ induce the assignment of x_1, x_{10}, x_{13} and columns 1, 10, 13 to part $\mathcal{P}_1$. Note that part $\mathcal{P}_1$ already contains rows 1, 10, 13 thus respecting the symmetric partitioning of **A**.

Consider an external net n_j with connectivity set Λ_j, where $\lambda_j = |\Lambda_j|$ and $\lambda_j > 1$. As all pins of net n_j lie in the parts in its connectivity set Λ_j, all rows (including row j by the consistency condition) which need x_j for



(a) (b) (c)

Figure 4: (a) A 16×16 structurally nonsymmetric matrix **A**. (b) Column-net representation$\mathcal{H}_{\mathcal{R}}$ of matrix **A** and 4-way partitioning Π of $\mathcal{H}_{\mathcal{R}}$. (c) 4-way rowwise decomposition of matrix $\mathbf{A}^{\Pi}$ obtained by permuting **A** according to the symmetric partitioning induced by Π.

inner-product computations are assigned to the parts (processors) in Λ_j. Hence, contribution $\lambda_j - 1$ of external net n_j to the cutsize according to (3.b) accurately models the amount of communication volume to incur during the parallel SpMxV computations because of x_j if x_j is assigned to any processor in Λ_j. Let $map[j] \in \Lambda_j$ denote the part and hence processor assignment for x_j corresponding to cut net n_j. In the column-net model together with the pre-communication scheme, cut net n_j indicates that processor $map[j]$ should send its local x_j to those processors in connectivity set Λ_j of net n_j except itself (i.e., to processors in the set $\Lambda_j - \{map[j]\}$). Hence, processor $map[j]$ should send its local x_j to $|\Lambda_j| - 1 = \lambda_j - 1$ distinct processors. As the consistency condition "$v_j \in n_j$" ensures that row j is already assigned to a part in Λ_j, symmetric partitioning of **A** can easily be maintained by assigning x_j hence permuting column j to the part which contains row j. In the 4-way decomposition shown in Fig. 4(b), external net n_5 (with $\Lambda_5 = \{\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3\}$) incurs the assignment of x_5 (hence permuting column 5) to part $\mathcal{P}_1$ since row 5 ($v_5 \in n_5$) is already assigned to part $\mathcal{P}_1$. The contribution $\lambda_5 - 1 = 2$ of net n_5 to the cutsize accurately models the communication volume to incur due to x_5, because processor P_1 should send x_5 to both processors P_2 and P_3 only once since $\Lambda_5 - \{map[5]\} = \Lambda_5 - \{P_1\} = \{P_2, P_3\}$.

In essence, in the column-net model, any partition Π of $\mathcal{H}_{\mathcal{R}}$ with $v_i \in \mathcal{P}_k$ can be safely decoded as assigning row i, y_i and x_i to processor P_k for rowwise decomposition. Similarly, in the row-net model, any partition Π of $\mathcal{H}_{\mathcal{C}}$ with $v_i \in \mathcal{P}_k$ can be safely decoded as assigning column i, x_i and y_i to processor P_k for columnwise decomposition. Thus, in the column-net and row-net models, minimizing the cutsize according to (3.b) corresponds to minimizing the actual volume of interprocessor communication during the pre and post communication phases, respectively. Maintaining the balance criterion (1) corresponds to maintaining the computational load balance during the local SpMxV computations. Figure 4(c) displays a permutation of the sample matrix given in Fig. 4(a) according to the symmetric partitioning induced by the 4-way decomposition shown in Fig. 4(b). As seen in Fig. 4(c), the actual communication volume for the given rowwise decomposition is 6 words since processor P_1 should send x_5 to both P_2 and P_3, P_2 should send x_{11} to P_4, P_3 should send x_7 to P_1, and P_4 should send x_{12} to both P_2 and P_3. As seen in Fig. 4(b), external nets n_5, n_7, n_{11} and n_{12} contribute 2, 1, 1 and 2 to the cutsize since $\lambda_5 = 3$, $\lambda_7 = 2$, $\lambda_{11} = 2$ and $\lambda_{12} = 3$, respectively. Hence, the cutsize of the 4-way decomposition given in Fig. 4(b) is 6, thus leading to the accurate modeling of the communication requirement. Note that the graph model will estimate the total communication volume as 13 words for the 4-way decomposition given in Fig. 4(c) since the total number of nonzeros in the off-diagonal blocks is 13. As seen in Fig. 4(c), each processor is assigned 12 nonzeros thus achieving perfect computational load balance.

In matrix theoretical view, let $\mathbf{A}^{\Pi}$ denote a permuted version of matrix **A** according to the symmetric partitioning induced by a partition Π of $\mathcal{H}_{\mathcal{R}}$ in the column-net model. Each cut-net n_j with connectivity set Λ_j and $map[j] = \mathcal{P}_\ell$ corresponds to column j of **A** containing nonzeros in λ_j distinct blocks ($\mathbf{A}^{\Pi}_{k\ell}$, for $\mathcal{P}_k \in \Lambda_j$) of matrix $\mathbf{A}^{\Pi}$. Since connectivity set Λ_j of net n_j is guaranteed to contain part $map[j]$, column j contains nonzeros in $\lambda_j - 1$ distinct off-diagonal blocks of $\mathbf{A}^{\Pi}$. Note that multiple nonzeros of column j in a particular off-diagonal block contributes only one to connectivity λ_j of net n_j by definition of λ_j. So, the cutsize of a partition Π of $\mathcal{H}_R$ is equal to the number of nonzero column segments in the off-diagonal blocks of matrix $\mathbf{A}^{\Pi}$. For example, external net n_5 with $\Lambda_5 = \{\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3\}$ and $map[5] = \mathcal{P}_1$ in Fig. 4(b) indicates that column 5 has nonzeros in two off-diagonal blocks $\mathbf{A}^{\Pi}_{2,1}$ and $\mathbf{A}^{\Pi}_{3,1}$ as seen in Fig. 4(c). As also seen in Fig. 4(c), the number of nonzero column segments in the off-diagonal blocks of matrix $\mathbf{A}^{\Pi}$ is 6 which is equal to the cutsize of partition Π shown in Fig. 4(b). Hence, the column-net model tries to achieve a symmetric permutation which minimizes the total number of nonzero column segments in the off-diagonal blocks for the pre-communication scheme. Similarly, the row-net model tries to achieve a symmetric permutation which minimizes the total number of nonzero row segments in the off-diagonal blocks for the post-communication scheme.

Nonzero diagonal entries automatically satisfy the condition "$v_j \in n_j$ for each net n_j", thus enabling both accurate representation of communication requirement and symmetric partitioning of **A**. A nonzero diagonal entry a_{jj} already implies that net n_j contains vertex v_j as its pin. If however some diagonal entries of the given matrix are zeros then the consistency of the proposed column-net model is easily maintained by simply adding rows, which do not contain diagonal entries, to the pin lists of the respective column nets. That is, if $a_{jj} = 0$ then vertex v_j (row j) is added to the pin list $pins[n_j]$ of net n_j and net n_j is added to the net list $nets[v_j]$ of vertex v_j. These pin additions do not affect the computational weight assignments of the vertices. That is, weight w_j of vertex v_j in $\mathcal{H}_{\mathcal{R}}$ becomes equal to either d_j or $d_j - 1$ depending on whether $a_{jj} \neq 0$ or $a_{jj} = 0$, respectively. The consistency of the row-net model is preserved in a dual manner.

4 DECOMPOSITION HEURISTICS

Kernighan-Lin (KL) based heuristics are widely used for graph/hypergraph partitioning because of their short run-times and good quality results. The KL algorithm is an iterative improvement heuristic originally proposed for graph bipartitioning [25]. The KL algorithm, starting from an initial bipartition, 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 the hypergraph bipartitioning problem by Schweikert-Kernighan [38]. Fiduccia-Mattheyses (FM) [10] introduced a faster implementation of the 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 lists, reduced the time complexity of a single pass of the 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.

The performance of the FM algorithm deteriorates for large and very sparse graphs/hypergraphs. Here, sparsity of graphs and hypergraphs refer to their average vertex degrees. Furthermore, the solution quality of FM is not *stable* (*predictable*), i.e., average FM solution is significantly worse than the best FM solution, which is a common weakness of the move-based iterative improvement approaches. Random multi-start approach is used in VLSI layout design to alleviate this problem by running the FM algorithm many times starting from random initial partitions to return the best solution found [1]. However, this approach is not viable in parallel computing since decomposition is a preprocessing overhead introduced to increase the efficiency of the underlying parallel algorithm/program. Most users will rely on one run of the decomposition heuristic, so the quality of the decomposition tool depends equally on the worst and average decompositions than on just the best decomposition.

These considerations have motivated the *two–phase* application of the move-based algorithms in hypergraph partitioning [12]. In this approach, a clustering is performed on the original hypergraph $\mathcal{H}_0$ to induce a coarser hypergraph $\mathcal{H}_1$. Clustering corresponds to coalescing highly interacting vertices to supernodes as a preprocessing to FM. Then, FM is run on $\mathcal{H}_1$ to find a bipartition Π_1, and this bipartition is projected back to a bipartition Π_0 of $\mathcal{H}_0$. Finally, FM is re-run on $\mathcal{H}_0$ using Π_0 as an initial solution. Recently, the two–phase approach has been extended to *multilevel* approaches [4, 13, 21] leading to successful graph partitioning tools Chaco [14] and MeTiS [22]. These multilevel heuristics consist of 3 phases: *coarsening*, *initial partitioning* and *uncoarsening*. In the first phase, a multilevel clustering is applied starting from the original graph by adopting various matching heuristics until the number of vertices in the coarsened graph reduces below a predetermined threshold value. In the second phase, the coarsest graph is partitioned using various heuristics including FM. In the third phase, the partition found in the second phase is successively projected back towards the original graph by refining the projected partitions on the intermediate level uncoarser graphs using various heuristics including FM.

In this work, we exploit the multilevel partitioning schemes for the experimental verification of the proposed hypergraph models in two approaches. In the first approach, multilevel graph partitioning tool MeTiS is used as a black box by transforming hypergraphs to graphs using the randomized clique-net model proposed in [2]. In the second approach, we have implemented a multilevel hypergraph partitioning tool PaToH, and tested both PaToH and multilevel hypergraph partitioning tool hMeTiS [23, 24] which was released very recently.

4.1 Randomized Clique-Net Model for Graph Representation of Hypergraphs

In the clique-net transformation model, the vertex set of the target graph is equal to the vertex set of the given hypergraph with the same vertex weights. Each net of the given hypergraph is represented by a clique of vertices corresponding to its pins. That is, each net induces an edge between every pair of its pins. The multiple edges connecting each pair of vertices of the graph are contracted into a single edge of which cost is equal to the sum of the costs of the edges it represents. In the *standard* clique-net model [29], a uniform cost of $1/(s_i - 1)$ is assigned to every clique edge of net n_i with size s_i. Various other edge weighting functions are also proposed in the literature [1]. 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. Ideally, no matter how vertices of a net are partitioned, the contribution of a cut net to the cutsize should always be one in a bipartition. However, the deficiency of the clique-net model is that it is impossible to achieve such a *perfect* clique-net model [18]. Furthermore, the transformation may result in very large graphs since the number of clique edges induced by the nets increase quadratically with their sizes.

Recently, a randomized clique-net model implementation is proposed [2] which yields very promising results when used together with graph partitioning tool MeTiS. In this model, all nets of size larger than T are removed during the transformation. Furthermore, for each net n_i of size s_i, $F \times s_i$ random pairs of its pins (vertices) are selected and an edge with cost one is added to the graph for each selected pair of vertices. The multiple edges between each pair of vertices of the resulting graph are contracted into a single edge as mentioned earlier. In this scheme, the nets with size smaller than $2F+1$ (small nets) induce larger number of edges than the standard clique-net model, whereas the nets with size larger than $2F+1$ (large nets) induce smaller number of edges than the standard clique-net model. Considering the fact that MeTiS accepts integer edge costs for the input graph, this scheme has two nice features[1]. First, it simulates the uniform edge-weighting scheme of the standard clique-net model for small nets in a random manner since each clique edge (if induced) of a net n_i with size $s_i < 2F+1$ will be assigned an integer cost close to $2F/(s_i - 1)$ on the average. Second, it prevents the quadratic increase in the number of clique edges induced by large nets in the standard model since the number of clique edges induced by a net in this scheme is linear in the size of the net. In our implementation, we use the parameters $T = 50$ and $F = 5$ in accordance with the recommendations given in [2].

4.2 PaToH: A Multilevel Hypergraph Partitioning Tool

In this work, we exploit the successful multilevel methodology [4, 13, 21] proposed and implemented for graph partitioning [14, 22] to develop a new multilevel hypergraph partitioning tool, called PaToH (PaToH: **Pa**rtitioning **To**ols for **H**ypergraphs).

The data structures used to store hypergraphs in PaToH mainly consist of the following arrays. The *NETLST* array stores the net lists of the vertices. The *PINLST* array stores the pin lists of the nets. The size of both arrays is equal to the total number of pins in the hypergraph. Two auxiliary index arrays *VTXS* and *NETS* of sizes $|\mathcal{V}|+1$ and $|\mathcal{N}|+1$ hold the starting indices of the net lists and pin lists of the vertices and nets in the *NETLST* and *PINLST* arrays, respectively. In sparse matrix storage terminology, this scheme corresponds to storing the given matrix both in *Compressed Sparse Row (CSR)* and *Compressed Sparse Column (CSC)* formats [27] without storing the numerical data. In the column-net model proposed for rowwise decomposition, the *VTXS* and *NETLST* arrays correspond to the CSR storage scheme, and the *NETS* and *PINLST* arrays correspond to the CSC storage scheme.

[1] private communication with Alpert.

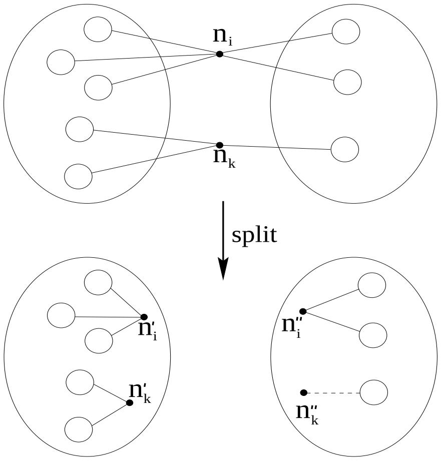


Figure 5: Cut-net splitting during recursive bisection.

This correspondence is dual in the row-net model proposed for columnwise decomposition.

The K-way graph/hypergraph partitioning problem is usually solved by recursive bisection. In this scheme, first a 2-way partition of $\mathcal{G}/\mathcal{H}$ is obtained, and then this bipartition is further partitioned in a recursive manner. After $\lg_2 K$ phases, graph $\mathcal{G}/\mathcal{H}$ is partitioned into K parts. PaToH achieves K-way hypergraph partitioning by recursive bisection for any K value (i.e., K is not restricted to be a power of 2).

The connectivity cutsize metric given in (3.b) needs special attention in K-way hypergraph partitioning by recursive bisection. Note that the cutsize metrics given in (3.a) and (3.b) become equivalent in hypergraph bisection. Consider a bipartition $\mathcal{V}_\mathcal{A}$ and $\mathcal{V}_\mathcal{B}$ of $\mathcal{V}$ obtained after a bisection step. It is clear that $\mathcal{V}_\mathcal{A}$ and $\mathcal{V}_\mathcal{B}$ and the internal nets of parts $\mathcal{A}$ and $\mathcal{B}$ will become the vertex and net sets of $\mathcal{H}_\mathcal{A}$ and $\mathcal{H}_\mathcal{B}$, respectively, for the following recursive bisection steps. Note that each cut net of this bipartition already contributes 1 to the total cutsize of the final K-way partition to be obtained by further recursive bisections. However, the further recursive bisections of $\mathcal{V}_\mathcal{A}$ and $\mathcal{V}_\mathcal{B}$ may increase the connectivity of these cut nets. In parallel SpMxV view, while each cut net already incurs the communication of a single word, these nets may induce additional communication because of the following recursive bisection steps. Hence, after every hypergraph bisection step, each cut net n_i is split into two pin-wise disjoint nets $n'_i = pins[n_i] \bigcap \mathcal{V}_\mathcal{A}$ and $n''_i = pins[n_i] \bigcap \mathcal{V}_\mathcal{B}$, and then these two nets are added to the net lists of $\mathcal{H}_\mathcal{A}$ and $\mathcal{H}_\mathcal{B}$ if $|n'_i| > 1$ and $|n''_i| > 1$, respectively. Note that the single-pin nets are discarded during the split operation since such nets cannot contribute to the cutsize in the following recursive bisection steps. Thus, the total cutsize according to (3.b) will become equal to the sum of the number of cut nets at every bisection step by using the above cut-net split method. Figure 5 illustrates two cut nets n_i and n_k in a bipartition, and their splits into nets n'_i, n''_i and n'_k, n''_k, respectively. Note that net n''_k becomes a single-pin net and it is discarded.

Similar to multilevel graph and hypergraph partitioning tools Chaco [14], MeTiS [22] and hMeTiS [24], the multilevel hypergraph bisection algorithm used in PaToH consists of 3 phases: coarsening, initial partitioning and uncoarsening. The following sections briefly summarize our multilevel bisection algorithm. Although PaToH works on weighted nets, we will assume unit cost nets both for the sake of simplicity of presentation and for the fact that all nets are assigned unit cost in the hypergraph representation of sparse matrices.

4.2.1 Coarsening Phase

In this phase, the given hypergraph $\mathcal{H} = \mathcal{H}_0 = (\mathcal{V}_0, \mathcal{N}_0)$ is coarsened into a sequence of smaller hypergraphs $\mathcal{H}_1 = (\mathcal{V}_1, \mathcal{N}_1)$, $\mathcal{H}_2 = (\mathcal{V}_2, \mathcal{N}_2)$, ..., $\mathcal{H}_m = (\mathcal{V}_m, \mathcal{N}_m)$ satisfying $|\mathcal{V}_0| > |\mathcal{V}_1| > |\mathcal{V}_2| > \ldots > |\mathcal{V}_m|$. This coarsening is achieved by coalescing disjoint subsets of vertices of hypergraph $\mathcal{H}_i$ into *multinodes* such that each multinode in $\mathcal{H}_i$ forms a single vertex of $\mathcal{H}_{i+1}$. The weight of each vertex of $\mathcal{H}_{i+1}$ becomes equal to the sum of its constituent vertices of the respective multinode in $\mathcal{H}_i$. The net set of each vertex of $\mathcal{H}_{i+1}$ becomes equal to the union of the net sets of the constituent vertices of the respective multinode in $\mathcal{H}_i$. Here, multiple pins of a net $n \in \mathcal{N}_i$ in a multinode cluster of $\mathcal{H}_i$ are contracted to a single pin of the respective net $n' \in \mathcal{N}_{i+1}$ of $\mathcal{H}_{i+1}$. Furthermore, the single-pin nets obtained during this contraction are discarded. Note that such single-pin nets correspond to the internal nets of the clustering performed on $\mathcal{H}_i$. The coarsening phase terminates when the number of vertices in the coarsened hypergraph reduces below 100 (i.e. $|\mathcal{V}_m| \leq 100$).

Clustering approaches can be classified as *agglomerative* and *hierarchical*. In the agglomerative clustering, new clusters are formed one at a time, whereas in the hierarchical clustering several new clusters may be formed simultaneously. In PaToH, we have implemented both randomized matching–based hierarchical clustering and randomized hierarchic–agglomerative clustering. The former and latter approaches will be abbreviated as matching–based clustering and agglomerative clustering, respectively.

The matching-based clustering works as follows. Vertices of $\mathcal{H}_i$ are visited in a random order. If a vertex $u \in \mathcal{V}_i$ has not been matched yet, one of its unmatched *adjacent* vertices is selected according to a criterion. If such a vertex v exists, we merge the matched pair u and v into a cluster. If there is no unmatched adjacent vertex of u, then vertex u remains unmatched, i.e., u remains as a singleton cluster. Here, two vertices u and v are said to be adjacent if they share at least one net, i.e., $nets[u] \cap nets[v] \neq \emptyset$. The selection criterion used in PaToH for matching chooses a vertex v with the highest connectivity value N_{uv}. Here, connectivity $N_{uv} = |nets[u] \cap nets[v]|$ refers to the number of shared nets between u and v. This matching-based scheme is referred to here as *Heavy Connectivity Matching* (*HCM*).

The matching-based clustering allows the clustering of only pairs of vertices in a level. In order to enable the clustering of more than two vertices at each level, we have implemented a randomized agglomerative clustering approach. In this scheme, each vertex u is assumed to constitute a singleton cluster $C_u = \{u\}$ at the beginning of each coarsening level. Then, vertices are visited in a random order. If a vertex u has already been clustered (i.e. $|C_u| > 1$) it is not considered for being the source of a new clustering. However, an unclustered vertex u can choose to join a multinode cluster as well as a singleton cluster. That is, all adjacent vertices of an unclustered vertex u are considered for selection according to a criterion. The selection of a vertex v adjacent to u corresponds to including vertex u to cluster C_v to grow a new multinode cluster $C_u = C_v = C_v \cup \{u\}$. Note that no singleton cluster remains at the end of this process as far as there exists no isolated vertex. The selection criterion used in PaToH for agglomerative clustering chooses a singleton or multinode cluster C_v with the highest $N_{u,C_v}/W_{u,C_v}$ value, where $N_{u,C_v} = |nets[u] \cap \bigcup_{x \in C_v} nets[x]|$ and W_{u,C_v} is the weight of the multinode cluster candidate

$$\mathbf{A}_0 = \left[\begin{array}{c|cccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline 1 & x & & x & & x & & & x \\ 2 & & x & x & & & x & & x \\ 3 & x & x & x & & x & x & & \\ 4 & & & & x & x & & & \\ 5 & x & & & x & x & & & \\ 6 & x & & & x & & x & x & x \\ 7 & & & & x & & x & x & \\ 8 & x & & & & & & & x \end{array}\right] \qquad \mathbf{A}_1^{HCM} = \left[\begin{array}{c|cccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline 1,3 & x & x & x & & x & x & & x \\ 2,6 & x & x & x & x & & x & x & x \\ 4,5 & x & & & x & x & & & \\ 7 & & & & x & & x & x & \\ 8 & x & & & & & & & x \end{array}\right]$$

$$\mathbf{A}_1^{HCC} = \left[\begin{array}{c|cccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline 1,2,3 & x & x & x & & x & x & & x \\ 4,5 & x & & & x & x & & & \\ 6,7,8 & x & & & x & & x & x & x \end{array}\right] = \left[\begin{array}{c|ccccc} & 1 & 4 & 5 & 6 & 8 \\ \hline 1,2,3 & x & & x & x & x \\ 4,5 & x & x & x & & \\ 6,7,8 & x & x & & x & x \end{array}\right]$$

Figure 6: Matching-based clustering $\mathbf{A}_1^{HCM}$ and agglomerative clustering $\mathbf{A}_1^{HCC}$ of the rows of matrix $\mathbf{A}_0$.

$\{u\} \cup C_v$. The division of N_{u,C_v} by W_{u,C_v} is an effort for avoiding the polarization towards very large clusters. This agglomerative clustering scheme is referred to here as *Heavy Connectivity Clustering* (*HCC*).

The objective in both HCM and HCC is to find highly connected vertex clusters. Connectivity values N_{uv} and N_{u,C_v} used for selection serve this objective. Note that N_{uv} (N_{u,C_v}) also denotes the lower bound in the amount of decrease in the number of pins because of the pin contractions to be performed when u joins v (C_v). Recall that there might be additional decrease in the number of pins because of single-pin nets that may occur after clustering. Hence, the connectivity metric is also an effort towards minimizing the complexity of the following coarsening levels, partitioning phase and refinement phase since the size of a hypergraph is equal to the number of its pins.

In rowwise matrix decomposition context (i.e. column-net model), the connectivity metric corresponds to the number of common column indices between two rows or row groups. Hence, both HCM and HCC try to combine rows or row groups with similar 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 the row-net model. Figure 6 illustrates a single level coarsening of an 8×8 sample matrix $\mathbf{A}_0$ in the column-net model using HCM and HCC. The original decimal ordering of the rows is assumed to be the random vertex visit order. As seen in Fig. 6, HCM matches row pairs $\{1,3\}$, $\{2,6\}$ and $\{4,5\}$ with the connectivity values of 3, 2 and 2, respectively. Note that the total number of nonzeros of $\mathbf{A}_0$ reduces from 28 to 21 in $\mathbf{A}_1^{HCM}$ after clustering. This difference is equal to the sum $3{+}2{+}2=7$ of the connectivity values of the matched row-vertex pairs since pin contractions do not lead to any single-pin nets. As seen in Fig. 6, HCC constructs three clusters $\{1,2,3\}$, $\{4,5\}$ and $\{6,7,8\}$ through the clustering sequence of $\{1,3\}$, $\{1,2,3\}$, $\{4,5\}$, $\{6,7\}$ and $\{6,7,8\}$ with the connectivity values of 3, 4, 2, 3 and 2, respectively. Note that pin contractions lead to three single-pin nets n_2, n_3 and n_7, thus columns 2, 3 and 7 are removed. As also seen in Fig. 6, although rows 7 and 8 remain unmatched in HCM, every row is involved in at least one clustering in HCC.

Both HCM and HCC necessitate scanning the pin lists of all nets in the net list of the source vertex to find its adjacent vertices for matching and clustering. In the column-net (row-net) model, the total cost of these scan operations can be as expensive as the total number of multiply and add operations which lead to nonzero entries in the computation of $\mathbf{A}\mathbf{A}^T$ ($\mathbf{A}^T\mathbf{A}$). In HCM, the key point to efficient implementation is to move the matched vertices encountered during the scan of the pin list of a net to the end of its pin list through a simple swap operation. This scheme avoids the re-visits of the matched vertices during the following matching operations at that level. Although this scheme requires an additional index array to maintain the temporary tail indices of the pin lists, it achieves substantial decrease in the run-time of the coarsening phase. Unfortunately, this simple yet effective scheme cannot be fully used in HCC. Since a singleton vertex can select a multinode cluster, the re-visits of the clustered vertices are partially avoided by maintaining only a single vertex to represent the multinode cluster in the pin-list of each net connected to the cluster, through simple swap operations. Through the use of these efficient implementation schemes the total cost of the scan operations in the column-net (row-net) model can be as low as the total number of nonzeros in $\mathbf{AA}^T$ ($\mathbf{A}^T\mathbf{A}$). In order to maintain this cost within reasonable limits, all nets of size greater than $4s_{avg}$ are not considered in a bipartitioning step, where s_{avg} denotes the average net size of the hypergraph to be partitioned in that step. Note that such nets can be reconsidered during the further levels of recursion because of net splitting.

The cluster growing operation in HCC requires disjoint-set operations for maintaining the representatives of the clusters, where the union operations are restricted to the union of a singleton source cluster with a singleton or a multinode target cluster. This restriction is exploited by always choosing the representative of the target cluster as the representative of the new cluster. Hence, it is sufficient to update the representative pointer of only the singleton source cluster joining to a multinode target cluster. Therefore, each disjoint-set operation required in this scheme is performed in $O(1)$ time.

4.2.2 Initial Partitioning Phase

The goal in this phase is to find a bipartition on the coarsest hypergraph $\mathcal{H}_m$. In PaToH, we use *Greedy Hypergraph Growing (GHG)* algorithm for bisecting $\mathcal{H}_m$. This algorithm can be considered as an extension of the GGGP algorithm used in MeTiS to hypergraphs. In GHG, we grow a cluster around a randomly selected vertex. During the coarse of the algorithm, the selected and unselected vertices induce a bipartition on $\mathcal{H}_m$. The unselected vertices connected to the growing cluster are inserted into a priority queue according to their FM gains. Here, the gain of an unselected vertex corresponds to the decrease in the cutsize of the current bipartition if the vertex moves to the growing cluster. Then, a vertex with the highest gain is selected from the priority queue. After a vertex moves to the growing cluster, the gains of its unselected adjacent vertices which are currently in the priority queue are updated and those not in the priority queue are inserted. This cluster growing operation continues until a predetermined bipartition balance criterion is reached. As also mentioned in MeTiS, the quality of this algorithm is sensitive to the choice of the initial random vertex. Since the coarsest hypergraph $\mathcal{H}_m$ is small, we run GHG 4 times starting from different random vertices and select the best bipartition for refinement during the uncoarsening phase.

4.2.3 Uncoarsening Phase

At each level i (for $i = m, m-1, \ldots, 1$), bipartition Π_i found on $\mathcal{H}_i$ is projected back to a bipartition Π_{i-1} on $\mathcal{H}_{i-1}$. The constituent vertices of each multinode in $\mathcal{H}_{i-1}$ is assigned to the part of the respective vertex in

$\mathcal{H}_i$. Obviously, Π_{i-1} of $\mathcal{H}_{i-1}$ has the same cutsize with Π_i of $\mathcal{H}_i$. Then, we refine this bipartition by running a *Boundary FM (BFM)* hypergraph bipartitioning algorithm on $\mathcal{H}_{i-1}$ starting from initial bipartition Π_{i-1}. BFM moves only the boundary vertices from the overloaded part to the under-loaded part, where a vertex is said to be a boundary vertex if it is connected to an at least one cut net.

BFM requires maintaining the *pin-connectivity* of each net for both initial gain computations and gain updates. The pin-connectivity $\sigma_k[n] = |n \cap \mathcal{P}_k|$ of a net n to a part $\mathcal{P}_k$ denotes the number of pins of net n that lie in part $\mathcal{P}_k$, for $k = 1, 2$. In order to avoid the scan of the pin lists of all nets, we adopt an efficient scheme to initialize the σ values for the first BFM pass in a level. It is clear that initial bipartition Π_{i-1} of $\mathcal{H}_{i-1}$ has the same cut-net set with Π_i of $\mathcal{H}_i$. Hence, we scan only the pin lists of the cut nets of Π_{i-1} to initialize their σ values. For each other net n, $\sigma_1[n]$ and $\sigma_2[n]$ values are easily initialized as $\sigma_1[n] = s_n$ and $\sigma_2[n] = 0$ if net n is internal to part $\mathcal{P}_1$, and $\sigma_1[n] = 0$ and $\sigma_2[n] = s_n$ otherwise. After initializing the gain value of each vertex v as $g[v] = -d_v$, we exploit σ values as follows. We re-scan the pin list of each external net n and update the gain value of each vertex $v \in pins[n]$ as $g[v] = g[v] + 2$ or $g[v] = g[v] + 1$ depending on whether net n is *critical* to the part containing v or not, respectively. An external net n is said to be critical to a part k if $\sigma_k[n] = 1$ so that moving the single vertex of net n that lies in that part to the other part removes net n from the cut. Note that two-pin cut nets are critical to both parts. The vertices visited while scanning the pin-lists of the external nets are identified as boundary vertices and only these vertices are inserted into the priority queue according to their computed gains.

In each pass of the BFM algorithm, a sequence of unmoved vertices with the highest gains are selected to move to the other part. As in the original FM algorithm, a vertex move necessitates gain updates of its adjacent vertices. However, in the BFM algorithm, some of the adjacent vertices of the moved vertex may not be in the priority queue, because they may not be boundary vertices before the move. Hence, such vertices which become boundary vertices after the move are inserted into the priority queue according to their updated gain values. The refinement process within a pass terminates when no *feasible* move remains or the sequence of last *max*$\{50,\ 0.001|\mathcal{V}_i|\}$ moves does not yield a decrease in the total cutsize. A move is said to be feasible if it does not disturb the load balance criterion (1) with $K = 2$. At the end of a BFM pass, we have a sequence of tentative vertex moves and their respective gains. We then construct from this sequence the maximum prefix subsequence of moves with the maximum prefix sum which incurs the maximum decrease in the cutsize. The permanent realization of the moves in this maximum prefix subsequence is efficiently achieved by rolling back the remaining moves at the end of the overall sequence. The initial gain computations for the following pass in a level is achieved through this rollback. The overall refinement process in a level terminates if the maximum prefix sum of a pass is not positive. In the current implementation of PaToH, at most 2 BFM passes are allowed at each level of the uncoarsening phase.

5 EXPERIMENTAL RESULTS

We have tested the validity of the proposed hypergraph models by running MeTiS on the graphs obtained by randomized clique-net transformation, and running PaToH and hMeTiS directly on the hypergraphs for the decompositions of various realistic sparse test matrices arising in different application domains. These decomposition results are compared with the decompositions obtained by running MeTiS using the standard and proposed graph models for the symmetric and nonsymmetric test matrices, respectively. The most recent version (Version 3.0) of MeTiS [22] was used in the experiments. As both hMeTiS and PaToH achieve K-way partitioning through recursive bisection, recursive MeTiS (pMeTiS) was used for the sake of a fair comparison. Another reason for using pMeTiS is that direct K-way partitioning version of MeTiS (kMeTiS) produces 9% worse partitions than pMeTiS in the decomposition of the nonsymmetric test matrices, although it is 2.5 times faster, on the average. pMeTiS was run with the default parameters: sorted heavy-edge matching, region growing and early-exit boundary FM refinement for coarsening, initial partitioning and uncoarsening phases, respectively. The current version (Version 1.0.2) of hMeTiS [24] was run with the parameters: greedy first-choice scheme (GFC) and early-exit FM refinement (EE-FM) for coarsening and uncoarsening phases, respectively. The V-cycle refinement scheme was not used, because in our experimentations it achieved at most 1% (much less on the average) better decompositions at the expense of approximately 3 times slower execution time (on the average) in the decomposition of the test matrices. The GFC scheme was found to be 28% faster than the other clustering schemes while producing slightly (1%–2%) better decompositions on the average. The EE-FM scheme was observed to be 30% faster than the other refinement schemes without any difference in the decomposition quality on the average.

Table I illustrates the properties of the test matrices listed in the order of increasing number of nonzeros. In this table, the "description" column displays both the nature and the source of each test matrix. The sparsity patterns of the Linear Programming matrices used as symmetric test matrices are obtained by multiplying the respective rectangular constraint matrices with their transposes. In Table I, the total number of nonzeros of a matrix also denotes the total number of pins in both column-net and row-net models. The minimum and maximum number of nonzeros per row (column) of a matrix correspond to the minimum and maximum vertex degree (net size) in the column-net model, respectively. Similarly, the standard deviation *std* and coefficient of variation *cov* values of nonzeros per row (column) of a matrix correspond to the *std* and *cov* values of vertex degree (net size) in the column-net model, respectively. Dual correspondences hold for the row-net model.

All experiments were carried out on a workstation equipped with a 133 MHz PowerPC processor with 512-Kbyte external cache and 64 Mbytes of memory. We have tested $K = 8$, 16, 32 and 64 way decompositions of every test matrix. For a specific K value, K-way decomposition of a test matrix constitutes a decomposition instance. pMeTiS, hMeTiS and PaToH were run 50 times starting from different random seeds for each decomposition instance. The average performance results are displayed in Tables II–IV and Figs. 7–9 for each decomposition instance. The percent load imbalance values are below 3% for all decomposition results displayed in these figures, where percent imbalance ratio is defined as $100 \times (W_{max} - W_{avg})/W_{avg}$.

Table II displays the decomposition performance of the proposed hypergraph models together with the standard graph model in the rowwise/columnwise decomposition of the symmetric test matrices. Note that the rowwise and columnwise decomposition problems become equivalent for symmetric matrices. Tables III and IV display the decomposition performance of the proposed column-net and row-net hypergraph models together with the proposed graph models in the rowwise and columnwise decompositions of the nonsymmetric test matrices, respectively. Due to lack of space, the decomposition performance results for the clique-net approach are not displayed in Tables II–IV, instead they are summarized in Table V. Although the main objective of this work is the minimization of the total communication volume, the results for the other performance metrics such as the maximum volume, average number and maximum number of messages handled by a single processor are also displayed in Tables II–IV. Note that the maximum volume and maximum number of messages determine the concurrent communication volume and concurrent number of messages, respectively, under the assumption that no congestion occurs in the network.

As seen in Tables II–IV, the proposed hypergraph models produce substantially better partitions than the graph model at each decomposition instance in terms of total communication volume cost. In the symmetric test matrices, the hypergraph model produces 7%–48% better partitions than the graph model (see Table II). In the nonsymmetric test matrices, the hypergraph models produce 12%–63% and 9%–56% better partitions than the graph models in the rowwise (see Table III) and columnwise (see Table IV) decompositions, respectively. As seen in Tables II–IV, there is no clear winner between hMeTiS and PaToH in terms of decomposition quality. In some matrices hMeTiS produces slightly better partitions than PaToH, whereas the situation is the other way round in some other matrices. As seen in Tables II and III, there is also no clear winner between clustering schemes HCM and HCC in PaToH. However, as seen in Table IV, PaToH-HCC produces slightly better partitions than PaToH-HCM in all columnwise decomposition instances for the nonsymmetric test matrices.

Tables II–IV show that the performance gap between the graph and hypergraph models in terms of the total communication volume costs is preserved by almost the same amounts in terms of the concurrent communication volume costs. For example, in the decomposition of the symmetric test matrices, the hypergraph model using PaToH-HCM incurs 30% less total communication volume than the graph model while incurring 28% less concurrent communication volume, on the overall average. In the columnwise decomposition of the nonsymmetric test matrices, PaToH-HCM incurs 35% less total communication volume than the graph model while incurring 37% less concurrent communication volume, on the overall average.

Although the hypergraph models perform better than the graph models in terms of number of messages, the performance gap is not as large as in the communication volume metrics. However, the performance gap increases with increasing K. As seen in Table II, in the 64-way decomposition of the symmetric test matrices, the hypergraph model using PaToH-HCC incurs 32% and 10% less total and concurrent number of messages than the graph model, respectively. As seen in Table III, in the rowwise decomposition of the nonsymmetric test matrices, PaToH-HCC incurs 32% and 26% less total and concurrent number of messages than the graph model, respectively.

The performance comparison of the graph/hypergraph partitioning based 1D decomposition schemes with the conventional algorithms based on 1D and 2D [15, 30] decomposition schemes is as follows. As mentioned earlier, in K-way decompositions of $m \times m$ matrices, the conventional 1D and 2D schemes incur the total communication volume of $(K-1)m$ and $2(\sqrt{K}-1)m$ words, respectively. For example, in 64-way decompositions, the conventional 1D and 2D schemes incur the total communication volumes of $63m$ and $14m$ words, respectively. As seen at the bottom of Tables II and III, PaToH-HCC reduces the total communication volume to $1.91m$ and $0.90m$ words in the 1D 64-way decomposition of the symmetric and nonsymmetric test matrices, respectively, on the overall average. In 64-way decompositions, the conventional 1D and 2D schemes incur the concurrent communication volumes of approximately m and $0.22m$ words, respectively. As seen in Tables II and III, PaToH-HCC reduces the concurrent communication volume to $0.052m$ and $0.025m$ words in the 1D 64-way decomposition of the symmetric and nonsymmetric test matrices, respectively, on the overall average.

Figure 7 illustrates the relative run-time performance of the proposed hypergraph model compared to the standard graph model in the rowwise/columnwise decomposition of the symmetric test matrices. Figures 8 and 9 display the relative run-time performance of the column-net and row-net hypergraph models compared to the proposed graph models in the rowwise and columnwise decompositions of the nonsymmetric test matrices, respectively. In Figs. 7–9, for each decomposition instance, we plot the ratios of the average execution times of the tools using the respective hypergraph model to that of pMeTiS using the respective graph model. The results displayed in Figs. 7–9 are obtained by assuming that the test matrix is given either in CSR or in CSC form which are commonly used for SpMxV computations. The standard graph model does not necessitate any preprocessing since CSR and CSC forms are equivalent in symmetric matrices and both of them correspond to the adjacency list representation of the standard graph model. However, in nonsymmetric matrices, construction of the proposed graph model requires some amount of preprocessing time, although we have implemented a very efficient construction code which totally avoids index search. Thus, the execution time averages of the graph models for the nonsymmetric test matrices include this preprocessing time. The preprocessing time constitutes approximately 3% of the total execution time on the overall average. In the clique-net model, transforming the hypergraph representation of the given matrices to graphs using the randomized clique-net model introduces considerable amount of preprocessing time, despite the efficient implementation scheme we have adopted. Hence, the execution time averages of the clique-net model include this transformation time. The transformation time constitutes approximately 23% of the total execution time on the overall average. As mentioned earlier, the PaToH and hMeTiS tools use both CSR and CSC forms such that the construction of the other form from the given one is performed within the respective tool.

As seen in Figs. 7–9, the tools using the hypergraph models run slower than pMeTiS using the the graph models in most of the instances. The comparison of Fig. 7 with Figs. 8 and 9 shows that the gap between the run-time performances of the graph and hypergraph models is much less in the decomposition of the nonsymmetric test matrices than that of the symmetric test matrices. These experimental findings were expected, because the execution times of graph partitioning tool pMeTiS, and hypergraph partitioning tools hMeTiS and PaToH are proportional to the sizes of the graph and hypergraph, respectively. In the representation of an $m \times m$ square matrix with Z off-diagonal nonzeros, the graph models contain $|\mathcal{E}| = Z/2$ and $Z/2 < |\mathcal{E}| \leq Z$ edges for symmetric and nonsymmetric matrices, respectively. However, the hypergraph models contain $p = m + Z$ pins for both symmetric and nonsymmetric matrices. Hence, the size of the hypergraph representation of a matrix is always greater than the size of its graph representation, and this gap in the sizes decreases in favor of the hypergraph models in nonsymmetric matrices. Figure 9 displays an interesting behavior that pMeTiS using the clique-net model runs faster than pMeTiS using the graph model in the columnwise decomposition of 4 out of 9 nonsymmetric test matrices. In these 4 test matrices, the edge contractions during the hypergraph-to-graph transformation through randomized clique-net approach lead to less number of edges than the graph model.

As seen in Figs. 7–9, both PaToH-HCM and PaToH-HCC run considerably faster than hMeTiS in each decomposition instance. This situation can be most probably due to the design considerations of hMeTiS. hMeTiS mainly aims at partitioning VLSI circuits of which hypergraph representations are much more sparse than the hypergraph representations of the test matrices. In the comparison of the HCM and HCC clustering schemes of PaToH, PaToH-HCM runs slightly faster than PaToH-HCC in the decomposition of almost all test matrices except in the decomposition of symmetric matrices KEN-11 and KEN-13, and nonsymmetric matrices ONETONE1 and ONETONE2. As seen in Fig. 7, PaToH-HCM using the hypergraph model runs 1.47–2.93 times slower than pMeTiS using the graph model in the decomposition of the symmetric test matrices. As seen in Figs. 8 and 9, PaToH-HCM runs 1.04–1.63 times and 0.83–1.79 times slower than pMeTiS using the graph model in the rowwise and columnwise decomposition of the nonsymmetric test matrices, respectively. Note that PaToH-HCM runs 17%, 8% and 6% faster than pMeTiS using the graph model in the 8-way, 16-way and 32-way columnwise decompositions of nonsymmetric matrix LHR34, respectively. PaToH-HCM achieves 64-way rowwise decomposition of the largest test matrix BCSSTK32 containing 44.6K rows/columns and 1030K nonzeros in only 25.6 seconds, which is equal to the sequential execution time of multiplying matrix BCSSTK32 with a dense vector 73.5 times.

The relative performance results of the hypergraph models with respect to the graph models are summarized in Table V in terms of total communication volume and execution time by averaging over different K values. This table also displays the averages of the best and worst performance results of the tools using the hypergraph models. In Table V, the performance results for the hypergraph models are normalized with respect to those of pMeTiS using the graph models. In the symmetric test matrices, direct approaches PaToH and hMeTiS produce 30%–32% better partitions than pMeTiS using the graph model, whereas the clique-net approach produces 16% better partitions, on the overall average. In the nonsymmetric test matrices, the direct approaches achieve 34%–38% better decomposition quality than pMeTiS using the graph model, whereas the clique-net approach achieves 21%–24% better decomposition quality. As seen in Table V, the clique-net approach is faster than the direct approaches in the decomposition of the symmetric test matrices. However, PaToH-HCM achieves nearly equal run-time performance as pMeTiS using the clique-net approach in the decomposition of the nonsymmetric test matrices. It is interesting to note that the execution time of the clique-net approach relative to the graph model decreases with increasing number of processors K. This is because of the fact that the percent preprocessing overhead due to the hypergraph-to-graph transformation in the total execution time of pMeTiS using the clique-net approach decreases with increasing K.

As seen in Table V, hMeTiS produces slightly (2%) better partitions at the expense of considerably larger execution time in the decomposition of the symmetric test matrices. However, PaToH-HCM achieves the same decomposition quality as hMeTiS for the nonsymmetric test matrices, whereas PaToH-HCC achieves slightly (2%–3%) better decomposition quality. In the decomposition of the nonsymmetric test matrices, although PaToH-HCC performs slightly better than PaToH-HCM in terms of decomposition quality, it is 13%–14% slower.

In the symmetric test matrices, the use of the proposed hypergraph model instead of the graph model achieves 30% decrease in the communication volume requirement of a single parallel SpMxV computation at the expense of 130% increase in the decomposition time by using PaToH-HCM for hypergraph partitioning. In the nonsymmetric test matrices, the use of the proposed hypergraph models instead of the graph model achieves 34%–35% decrease in the communication volume requirement of a single parallel SpMxV computation at the expense of only 34%–39% increase in the decomposition time by using PaToH-HCM.

6 CONCLUSION

Two computational hypergraph models were proposed to decompose sparse matrices for minimizing communication volume while maintaining load balance during repeated parallel matrix-vector product computations. The proposed models enable the representation and hence the decomposition of structurally nonsymmetric matrices as well as structurally symmetric matrices. Furthermore, they introduce a much more accurate representation for the communication requirement than the standard computational graph model widely used in the literature for the parallelization of various scientific applications. The proposed models reduce the decomposition problem to the well-known hypergraph partitioning problem thus enabling the use of circuit partitioning heuristics widely used in VLSI design. The successful multilevel graph partitioning tool MeTiS was used for the experimental evaluation of the validity of the proposed hypergraph models through hypergraph-to-graph transformation using the randomized clique-net model. A successful multilevel hypergraph partitioning tool PaToH was also implemented, and both PaToH and recently released multilevel hypergraph partitioning tool hMeTiS were used for testing the validity of the proposed hypergraph models. Experimental results carried out on a wide range of sparse test matrices arising in different application domains confirmed the validity of the proposed hypergraph models. In the decomposition of the test matrices, the use of the proposed hypergraph models instead of the graph models achieved 30%-38% decrease in the communication volume requirement of a single parallel matrix-vector multiplication at the expense of only 34%–130% increase in the decomposition time by using PaToH, on the average. This work was also an effort towards showing that the computational hypergraph model is more powerful than the standard computational graph model as it provides a more versatile representation for the interactions among the atomic tasks of the computational domains.

References

[1] C. J. Alpert and A. B. Kahng, "Recent directions in netlist partitioning: A survey," *VLSI Journal*, vol. 19, no. 1-2, pp. 1–81, 1995.

[2] C. J. Alpert, L. W. Hagen, and A. B. Kahng, "A hybrid multilevel/genetic approach for circuit partitioning," tech. rep., UCLA Computer Science Department, 1996.

[3] C. Aykanat, F. Ozguner, F. Ercal, and P. Sadayappan, "Iterative algorithms for solution of large sparse systems of linear equations on hypercubes," *IEEE Transactions on Computers*, vol. 37, no. 12, pp. 1554–1567, Dec. 1988.

[4] T. Bui, and C. Jones, "A heuristic for reducing fill in sparse matrix factorization," in *Proc. 6th SIAM Conf. Parallel Processing for Scientific Computing*, pp. 445–452, 1993.

[5] T. Bultan and C. Aykanat, "A new mapping heuristic based on mean field annealing," *J. Parallel and Distributed Computing*, vol. 16, pp. 292–305, 1992.

[6] W. Camp, S. J. Plimpton, B. Hendrickson, and R. W. Leland, "Massively parallel methods for engineering and science problems," *Communication of ACM*, vol. 37, pp. 31–41, April 1994.

[7] 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*, vol. 38, no. 2, pp. 240–248, 1990.

[8] Ü. V. Çatalyürek and C. Aykanat, "Decomposing irregularly sparse matrices for parallel matrix-vector multiplications," in *Proc. 3rd Int. Workshop on Parallel Algorithms for Irregularly Structured Problems (IRREGULAR'96)*, pp. 175–181, 1996.

[9] I. S. Duff, R. Grimes, and J. Lewis, "Sparse matrix test problems," *ACM Transactions on Mathematical Software*, vol. 15, pp. 1–14, March 1989.

[10] 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*, pp. 175–181, 1982.

[11] M. Garey, D. Johnson, and L. Stockmeyer, "Some simplified NP-complete graph problems," *Theoretical Computer Science*, vol. 1, pp. 237–267, 1976.

[12] M. K. Goldberg, and M. Burstein, "Heuristic improvement techniques for bisection of vlsi networks," in *Proc. IEEE Intl. Conf. Computer Design*, pp. 122–125, 1983.

[13] B. Hendrickson and R. Leland, "A multilevel algorithm for partitioning graphs," tech. rep., Sandia National Laboratories, 1993.

[14] B. Hendrickson and R. Leland, *The Chaco user's guide, version 2.0*, tech. rep. SAND95-2344, Sandia National Laboratories, Alburquerque, NM, 87185, 1995.

[15] B. Hendrickson, R. Leland, and S. Plimpton, "An efficient parallel algorithm for matrix-vector multiplication," *Int. J. High Speed Computing*, vol. 7, no. 1, pp. 73–88, 1995.

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

[17] B. Hendrickson and T. G. Kolda "Partitioning rectangular and structurally nonsymmetric sparse matrices for parallel processing," submitted to *SIAM Journal on Scientific Computing*.

[18] E. Ihler, D. Wagner, and F. Wagner, "Modeling hypergraphs by graphs with the same mincut properties," *Information Processing Letters*, vol. 45, pp. 171–175, March 1993.

[19] IOWA Optimization Center, Linear programming problems, ftp://col.biz.uiowa.edu:pub/testprob/lp/gondzio.

[20] M. Kaddoura, C. W. Qu, and S. Ranka, “Partitioning unstructured computational graphs for nonuniform and adaptive environments,” *IEEE Parallel and Distributed Technology*, pp. 63–69, 1995.

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

[22] G. Karypis and V. Kumar, *MeTiS A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices Version 3.0*. University of Minnesota, Department of Comp. Sci. and Eng., Army HPC Research Center, Minneapolis, 1998.

[23] G. Karypis, V. Kumar, R. Aggarwal, and S. Shekhar, “Hypergraph partitioning using multilevel approach: applications in VLSI domain,” *IEEE Transactions on VLSI Systems*, to appear.

[24] G. Karypis, V. Kumar, R. Aggarwal, and S. Shekhar, *hMeTiS A Hypergraph Partitioning Package Version 1.0.1*. University of Minnesota, Department of Comp. Sci. and Eng., Army HPC Research Center, Minneapolis, 1998.

[25] B. W. Kernighan and S. Lin, “An efficient heuristic procedure for partitioning graphs,” *The Bell System Technical Journal*, vol. 49, pp. 291–307, Feb. 1970.

[26] T. G. Kolda, “Partitioning sparse rectangular matrices for parallel processing,” *Lecture Notes in Computer Science*, vol. 1457, pp. 68–79, 1998.

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

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

[29] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*. Chichester, U.K.: Wiley, 1990.

[30] J. G. Lewis and R. A. van de Geijn, “Distributed memory matrix-vector multiplication and conjugate gradient algorithms,” in *Proc. Supercomputing’93*, pp. 15–19, 1993.

[31] O. C. Martin and S. W. Otto, “Partitioning of unstructured meshes for load balancing,” *Concurrency: Practice and Experience*, vol. 7, no. 4, pp. 303–314, 1995.

[32] S. G. Nastea, O. Frieder, and T. El-Ghazawi, “Load-balanced sparse matrix-vector multiplication on parallel computers,” *J. Parallel and Distributed Computing*, vol. 46, pp. 439–458, 1997.

[33] A. T. Ogielski and W. Aielo, “Sparse matrix computations on parallel processor arrays,” *SIAM J. Scientific Comput.*, 1993.

[34] A. Pınar, Ü. V. Çatalyürek, C. Aykanat, and M. Pınar, “Decomposing linear programs for parallel solution,” *Lecture Notes in Computer Science*, vol. 1041, pp. 473–482, 1996.

[35] C. Pommerell, M. Annaratone, and W. Fichtner, “A set of new mapping and coloring heuristics for distributed-memory parallel processors,” *SIAM J. Scientific and Statistical Computing*, vol. 13, pp. 194–226, Jan. 1992.

[36] C.-W. Qu and S. Ranka, “Parallel incremental graph partitioning,” *IEEE Trans. Parallel and Distributed Systems*, vol. 8, no. 8, pp. 884–896, 1997.

[37] Y. Saad, K. Wu, and S. Petiton, “Sparse matrix computations on the CM-5,” in *Proc. 6th SIAM Conf. on Parallel Processing for Scientifical Computing*, 1993.

[38] 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*, pp. 57–62, 1972.

[39] T. Davis, University of Florida Sparse Matrix Collection, http://www.cise.ufl.edu/ davis/sparse/, NA Digest, vol. 92/96/97, no. 42/28/23, 1994/1996/1997.

Table I: Properties of test matrices.

matrix name	description	number	number of nonzeros										
		of	total	avg. per	per column				per row				
		rows/cols		row/col	min	max	std	cov	min	max	std	cov	
Structurally Symmetric Matrices													
SHERMAN3	[9] 3D finite difference grid	5005	20033	4.00	1	7	2.66	0.67	1	7	2.66	0.67	
KEN-11	[7] linear programming	14694	82454	5.61	2	243	14.54	2.59	2	243	14.54	2.59	
NL	[19] linear programming	7039	105089	14.93	1	361	28.48	1.91	1	361	28.48	1.91	
KEN-13	[7] linear programming	28632	161804	5.65	2	339	16.84	2.98	2	339	16.84	2.98	
CQ9	[19] linear programming	9278	221590	23.88	1	702	54.46	2.28	1	702	54.46	2.28	
CO9	[19] linear programming	10789	249205	23.10	1	707	52.17	2.26	1	707	52.17	2.26	
CRE-D	[7] linear programming	8926	372266	41.71	1	845	76.46	1.83	1	845	76.46	1.83	
CRE-B	[7] linear programming	9648	398806	41.34	1	904	74.69	1.81	1	904	74.69	1.81	
FINAN512	[39] stochastic programming	74752	615774	8.24	3	1449	20.00	2.43	3	1449	20.00	2.43	
Structurally Nonsymmetric Matrices													
GEMAT11	[9] optimal power flow	4929	38101	7.73	1	28	2.96	0.38	1	29	3.38	0.44	
LHR07	[39] light hydrocarbon recovery	7337	163716	22.31	1	64	26.19	1.17	2	37	16.00	0.72	
ONETONE2	[39] nonlinear analog circuit	36057	254595	7.06	2	34	5.13	0.73	2	66	6.67	0.94	
LHR14	[39] light hydrocarbon recovery	14270	321988	22.56	1	64	26.26	1.16	2	37	15.98	0.71	
ONETONE1	[39] nonlinear analog circuit	36057	368055	10.21	2	82	14.32	1.40	2	162	17.85	1.75	
LHR17	[39] light hydrocarbon recovery	17576	399500	22.73	1	64	26.32	1.16	2	37	15.96	0.70	
LHR34	[39] light hydrocarbon recovery	35152	799064	22.73	1	64	26.32	1.16	2	37	15.96	0.70	
BCSSTK32	[9] 3D stiffness matrix	44609	1029655	23.08	1	141	10.10	0.44	1	192	10.45	0.45	
BCSSTK30	[9] 3D stiffness matrix	28924	1036208	35.83	1	159	21.99	0.61	1	104	15.27	0.43	

Table II: Average communication requirements for rowwise/columnwise decomposition of structurally symmetric test matrices.

name	K	Graph Model				Hypergraph Model: Column-net Model $\equiv$ Row-net Model											
		pMeTiS				hMeTiS				PaToH-HCM				PaToH-HCC			
		# of mssgs per proc.		comm. volume		# of mssgs per proc.		comm. volume		# of mssgs per proc.		comm. volume		# of mssgs per proc.		comm. volume	
		avg	max	tot	max	avg	max	tot	max	avg	max	tot	max	avg	max	tot	max
SHERMAN3	8	3.6	4.9	0.20	0.033	3.6	5.0	0.17	0.029	3.4	4.9	0.16	0.030	3.3	4.8	0.16	0.030
	16	5.3	8.2	0.31	0.028	5.2	7.8	0.27	0.024	4.5	7.4	0.25	0.024	4.7	7.8	0.25	0.025
	32	6.5	11.0	0.46	0.021	6.7	10.9	0.39	0.018	5.7	10.1	0.37	0.019	5.9	10.5	0.37	0.019
	64	7.5	13.6	0.64	0.016	7.9	13.6	0.55	0.013	7.0	13.1	0.53	0.014	7.0	13.4	0.53	0.014
KEN-11	8	7.0	7.0	0.70	0.116	6.9	7.0	0.47	0.078	6.9	7.0	0.51	0.083	7.0	7.0	0.55	0.094
	16	13.8	15.0	0.92	0.080	12.4	15.0	0.57	0.047	12.8	15.0	0.59	0.046	13.7	15.0	0.66	0.057
	32	26.1	30.5	1.16	0.055	19.8	30.3	0.70	0.032	21.2	31.0	0.73	0.033	22.1	30.5	0.79	0.034
	64	40.9	54.9	1.44	0.038	30.1	58.6	0.90	0.024	32.1	60.4	0.92	0.025	30.1	54.2	0.96	0.025
NL	8	7.0	7.0	1.33	0.192	6.8	7.0	0.72	0.110	6.8	7.0	0.76	0.124	7.0	7.0	0.79	0.135
	16	15.0	15.0	1.71	0.147	13.5	15.0	0.99	0.085	13.2	15.0	1.05	0.097	13.7	15.0	1.14	0.101
	32	28.1	31.0	2.26	0.101	19.5	26.5	1.40	0.060	20.0	27.6	1.52	0.068	20.3	27.5	1.57	0.070
	64	38.2	59.1	3.06	0.073	24.4	39.3	2.08	0.045	26.4	40.5	2.20	0.048	26.0	42.9	2.23	0.050
KEN-13	8	7.0	7.0	0.75	0.120	7.0	7.0	0.47	0.070	7.0	7.0	0.48	0.075	6.9	7.0	0.48	0.076
	16	14.8	15.0	0.94	0.078	13.2	15.0	0.54	0.043	14.0	15.0	0.55	0.041	13.4	15.0	0.55	0.042
	32	29.2	31.0	1.16	0.051	22.7	31.0	0.64	0.029	22.8	31.0	0.63	0.025	21.8	31.0	0.63	0.027
	64	51.0	62.2	1.41	0.034	35.9	62.8	0.80	0.022	35.8	63.0	0.79	0.020	34.7	63.0	0.78	0.019
CQ9	8	7.0	7.0	1.11	0.173	7.0	7.0	0.65	0.104	7.0	7.0	0.71	0.154	6.9	7.0	0.71	0.166
	16	14.9	15.0	1.69	0.172	12.7	15.0	0.88	0.097	12.9	15.0	0.99	0.120	12.7	14.9	0.96	0.112
	32	21.8	30.7	2.42	0.148	18.6	26.6	1.36	0.075	18.0	27.0	1.47	0.086	17.6	26.9	1.40	0.082
	64	32.1	56.4	3.71	0.115	23.7	38.4	2.27	0.061	22.7	41.0	2.34	0.065	22.7	39.5	2.31	0.064
CO9	8	7.0	7.0	0.96	0.156	7.0	7.0	0.67	0.110	7.0	7.0	0.68	0.133	7.0	7.0	0.67	0.139
	16	14.8	15.0	1.51	0.157	12.4	14.9	0.87	0.091	12.7	14.9	0.94	0.110	12.7	14.9	0.92	0.107
	32	19.5	29.7	2.08	0.120	17.6	26.6	1.33	0.079	17.6	26.3	1.37	0.077	18.1	26.7	1.34	0.079
	64	29.9	52.3	3.14	0.093	21.7	37.3	2.13	0.061	21.8	38.8	2.16	0.059	21.9	38.6	2.14	0.062
CRE-D	8	7.0	7.0	1.81	0.292	6.9	7.0	1.39	0.226	6.4	7.0	1.33	0.214	6.2	7.0	1.25	0.208
	16	14.9	15.0	2.81	0.238	13.0	15.0	2.09	0.177	11.8	15.0	2.00	0.176	11.2	15.0	1.89	0.163
	32	28.7	31.0	4.13	0.188	21.3	31.0	2.97	0.136	19.3	31.0	2.89	0.133	18.4	31.0	2.73	0.124
	64	47.9	63.0	6.01	0.142	31.2	61.3	4.16	0.104	29.7	60.8	4.19	0.104	27.9	60.5	3.96	0.098
CRE-B	8	7.0	7.0	1.70	0.267	6.9	7.0	1.40	0.224	6.7	7.0	1.33	0.213	6.6	7.0	1.28	0.212
	16	14.8	15.0	2.62	0.230	13.4	15.0	2.07	0.177	12.2	15.0	2.01	0.175	12.2	15.0	1.95	0.180
	32	28.5	31.0	3.89	0.179	21.5	30.9	2.90	0.138	20.0	31.0	2.88	0.148	19.3	31.0	2.75	0.154
	64	46.6	63.0	5.72	0.136	31.3	61.4	4.07	0.111	30.0	61.7	4.12	0.121	28.3	61.5	3.93	0.125
FINAN512	8	2.9	4.3	0.13	0.047	2.8	4.2	0.11	0.045	3.0	4.6	0.12	0.047	3.4	5.6	0.12	0.047
	16	4.3	7.2	0.20	0.034	3.0	6.7	0.14	0.024	3.3	7.2	0.16	0.025	4.0	9.4	0.17	0.027
	32	6.3	13.6	0.27	0.020	3.4	13.2	0.18	0.015	4.2	13.8	0.21	0.016	4.7	17.3	0.22	0.017
	64	8.8	26.5	0.38	0.013	4.2	25.8	0.28	0.010	5.5	26.4	0.31	0.011	5.9	31.0	0.32	0.012
Averages over K																	
	8	6.2	6.5	0.97	0.155	6.1	6.5	0.67	0.111	6.0	6.5	0.68	0.119	6.0	6.6	0.67	0.123
	16	12.5	13.4	1.41	0.129	11.0	13.3	0.93	0.085	10.8	13.3	0.95	0.091	10.9	13.6	0.94	0.090
	32	21.6	26.6	1.98	0.098	16.8	25.2	1.32	0.065	16.5	25.4	1.34	0.067	16.5	25.8	1.31	0.067
	64	33.6	50.1	2.83	0.073	23.4	44.3	1.92	0.050	23.4	45.1	1.95	0.052	22.7	45.0	1.91	0.052

In the "# of mssgs" column, "avg" and "max" denote the average and maximum number of messages, respectively, handled by a single processor. In the "comm. volume" column, "tot" denotes the total communication volume, whereas "max" denotes the maximum communication volume handled by a single processor. Communication volume values (in terms of the number of words transmitted) are scaled by the number of rows/columns of the respective test matrices.

Table III: Average communication requirement for rowwise decomposition of structurally nonsymmetric test matrices.

name	K	Graph Model				Hypergraph Model: Column-net Model											
		pMeTiS				hMeTiS				PaToH-HCM				PaToH-HCC			
		# of mssgs per proc.		comm. volume		# of mssgs per proc.		comm. volume		# of mssgs per proc.		comm. volume		# of mssgs per proc.		comm. volume	
		avg	max	tot	max	avg	max	tot	max	avg	max	tot	max	avg	max	tot	max
GEMAT11	8	7.0	7.0	1.33	0.201	7.0	7.0	0.79	0.111	7.0	7.0	0.75	0.109	7.0	7.0	0.73	0.106
	16	15.0	15.0	1.85	0.144	14.8	15.0	1.00	0.071	14.7	15.0	0.96	0.070	14.6	15.0	0.93	0.067
	32	29.8	31.0	2.31	0.092	26.6	30.8	1.18	0.044	25.8	30.6	1.15	0.043	25.1	30.4	1.10	0.042
	64	47.7	58.8	2.71	0.056	34.3	46.7	1.33	0.026	33.5	46.2	1.32	0.026	31.9	44.2	1.27	0.025
LHR07	8	6.8	7.0	1.09	0.179	6.2	7.0	0.64	0.111	6.0	7.0	0.65	0.106	5.8	7.0	0.66	0.116
	16	13.0	15.0	1.52	0.130	10.3	13.9	0.93	0.089	9.7	13.8	0.91	0.081	9.2	13.1	0.90	0.083
	32	20.1	29.1	1.96	0.094	13.9	22.3	1.30	0.081	13.0	21.7	1.24	0.066	12.5	20.5	1.24	0.064
	64	24.4	44.8	2.49	0.079	16.8	33.5	1.84	0.077	15.6	30.0	1.65	0.056	15.9	30.7	1.64	0.059
ONETONE2	8	2.8	4.3	0.08	0.014	2.6	3.8	0.06	0.010	2.4	3.5	0.06	0.011	2.5	3.6	0.06	0.010
	16	4.9	7.5	0.17	0.015	4.9	7.3	0.11	0.010	4.7	6.9	0.12	0.011	4.7	6.8	0.12	0.011
	32	7.0	11.9	0.28	0.014	7.5	13.3	0.20	0.009	8.0	11.9	0.22	0.009	7.1	10.9	0.21	0.009
	64	9.4	18.6	0.39	0.011	10.1	20.1	0.29	0.007	10.7	17.2	0.31	0.008	9.4	15.8	0.31	0.008
LHR14	8	7.0	7.0	0.99	0.157	6.6	7.0	0.61	0.100	6.4	7.0	0.59	0.095	6.2	7.0	0.59	0.097
	16	14.0	15.0	1.33	0.116	11.4	14.6	0.84	0.074	10.3	13.5	0.81	0.071	10.0	13.6	0.82	0.072
	32	22.9	29.4	1.71	0.078	15.5	23.2	1.10	0.056	13.5	20.7	1.05	0.050	13.1	20.9	1.07	0.053
	64	29.9	48.6	2.14	0.054	18.1	31.5	1.44	0.048	15.4	27.5	1.34	0.040	15.6	29.0	1.36	0.041
ONETONE1	8	5.1	6.5	0.42	0.067	3.7	5.0	0.16	0.025	3.5	4.9	0.16	0.026	3.6	4.9	0.16	0.025
	16	8.5	11.8	0.59	0.050	7.9	10.4	0.29	0.023	7.6	9.8	0.30	0.026	7.8	10.1	0.29	0.024
	32	13.6	19.1	0.78	0.035	14.2	19.7	0.42	0.017	13.8	19.1	0.45	0.020	14.2	18.9	0.42	0.019
	64	18.7	28.9	0.97	0.025	22.0	33.0	0.57	0.013	19.3	29.2	0.61	0.016	19.8	29.7	0.56	0.015
LHR17	8	7.0	7.0	0.94	0.143	6.9	7.0	0.62	0.094	6.7	7.0	0.57	0.090	6.5	7.0	0.60	0.095
	16	14.3	15.0	1.28	0.110	12.4	14.8	0.82	0.068	11.0	13.8	0.77	0.066	10.8	13.7	0.80	0.068
	32	23.5	29.6	1.62	0.074	17.1	23.8	1.07	0.052	14.4	21.0	1.00	0.047	14.1	21.5	1.03	0.047
	64	30.3	46.9	2.04	0.048	19.6	33.0	1.38	0.041	16.4	29.4	1.29	0.036	16.0	30.3	1.30	0.036
LHR34	8	3.5	4.8	0.61	0.088	3.6	5.3	0.42	0.063	3.5	5.0	0.38	0.056	3.4	4.5	0.40	0.061
	16	7.3	9.5	0.95	0.075	7.3	10.1	0.62	0.049	7.0	9.7	0.57	0.046	6.8	8.8	0.60	0.050
	32	14.5	17.5	1.28	0.055	12.6	16.8	0.84	0.037	11.1	15.3	0.77	0.034	10.9	14.6	0.80	0.035
	64	23.7	30.6	1.63	0.038	17.2	24.9	1.08	0.027	14.6	22.7	1.00	0.025	14.3	22.5	1.03	0.025
BCSSTK32	8	3.5	5.4	0.07	0.015	3.7	5.7	0.05	0.012	3.5	5.4	0.05	0.013	3.6	5.5	0.05	0.012
	16	4.4	7.6	0.12	0.013	4.2	8.3	0.09	0.011	4.0	7.3	0.09	0.011	4.0	7.3	0.09	0.011
	32	5.1	9.4	0.20	0.011	4.7	10.6	0.14	0.008	4.7	9.6	0.15	0.009	4.6	9.7	0.14	0.008
	64	5.7	11.3	0.30	0.008	4.8	11.6	0.22	0.006	4.9	11.0	0.24	0.007	4.7	10.8	0.22	0.006
BCSSTK30	8	2.3	3.9	0.10	0.018	2.3	3.6	0.09	0.018	2.2	3.4	0.09	0.017	2.2	3.4	0.08	0.017
	16	3.7	6.3	0.21	0.022	3.3	5.4	0.18	0.018	3.3	5.6	0.18	0.018	3.3	5.6	0.16	0.017
	32	4.9	8.7	0.36	0.019	4.4	7.9	0.29	0.015	4.6	8.0	0.31	0.016	4.4	7.8	0.28	0.014
	64	5.8	11.3	0.57	0.016	5.3	10.6	0.45	0.013	5.6	10.3	0.48	0.013	5.3	10.0	0.45	0.012
Averages over K																	
	8	5.0	5.9	0.63	0.098	4.7	5.7	0.38	0.060	4.6	5.6	0.37	0.058	4.5	5.5	0.37	0.060
	16	9.5	11.4	0.89	0.075	8.5	11.1	0.54	0.046	8.0	10.6	0.53	0.045	7.9	10.4	0.52	0.045
	32	15.7	20.6	1.17	0.052	12.9	18.7	0.73	0.036	12.1	17.5	0.70	0.033	11.8	17.3	0.70	0.032
	64	21.7	33.3	1.47	0.037	16.5	27.2	0.96	0.029	15.1	24.8	0.92	0.025	14.8	24.8	0.90	0.025

In the "# of mssgs" column, "avg" and "max" denote the average and maximum number of messages, respectively, handled by a single processor. In the "comm. volume" column, "tot" denotes the total communication volume, whereas "max" denotes the maximum communication volume handled by a single processor. Communication volume values (in terms of the number of words transmitted) are scaled by the number of rows/columns of the respective test matrices.

Table IV: Average communication requirements for columnwise decomposition of structurally nonsymmetric test matrices.

name	K	Graph Model				Hypergraph Model: Row-net Model											
		pMeTiS				hMeTiS				PaToH-HCM				PaToH-HCC			
		# of mssgs per proc.		comm. volume		# of mssgs per proc.		comm. volume		# of mssgs per proc.		comm. volume		# of mssgs per proc.		comm. volume	
		avg	max	tot	max	avg	max	tot	max	avg	max	tot	max	avg	max	tot	max
GEMAT11	8	7.0	7.0	1.44	0.213	7.0	7.0	0.75	0.108	7.0	7.0	0.76	0.110	7.0	7.0	0.72	0.108
	16	15.0	15.0	1.98	0.145	14.7	15.0	0.95	0.071	14.7	15.0	0.97	0.072	14.6	15.0	0.93	0.069
	32	29.9	31.0	2.46	0.091	25.6	30.0	1.13	0.043	25.9	30.3	1.15	0.043	25.0	29.9	1.10	0.042
	64	47.9	58.5	2.85	0.056	32.7	43.9	1.28	0.026	33.6	45.3	1.33	0.026	31.6	43.8	1.27	0.025
LHR07	8	6.9	7.0	1.10	0.188	6.5	7.0	0.75	0.123	6.4	7.0	0.67	0.107	6.4	7.0	0.66	0.105
	16	12.5	15.0	1.54	0.141	11.1	15.0	1.10	0.094	10.6	15.0	0.96	0.081	10.8	15.0	0.95	0.081
	32	19.3	30.3	2.05	0.112	16.4	28.7	1.52	0.068	15.1	29.5	1.32	0.059	15.6	29.0	1.31	0.059
	64	23.5	56.7	2.60	0.088	22.0	39.2	2.03	0.050	19.7	40.5	1.76	0.042	19.8	41.2	1.74	0.042
ONETONE2	8	2.6	3.8	0.09	0.017	2.4	3.2	0.07	0.012	2.2	3.1	0.08	0.013	3.1	4.5	0.08	0.013
	16	4.8	7.4	0.20	0.019	4.7	6.6	0.13	0.012	4.6	6.2	0.16	0.014	5.4	8.7	0.15	0.014
	32	7.5	12.7	0.34	0.016	7.6	11.2	0.24	0.010	7.6	11.1	0.27	0.011	8.3	14.8	0.25	0.011
	64	10.2	21.4	0.46	0.013	9.6	15.8	0.33	0.008	10.5	16.4	0.35	0.008	10.4	23.5	0.34	0.009
LHR14	8	7.0	7.0	1.05	0.168	6.6	7.0	0.67	0.109	6.6	7.0	0.61	0.096	6.7	7.0	0.61	0.096
	16	13.9	15.0	1.43	0.123	11.4	14.7	0.95	0.077	11.6	15.0	0.85	0.069	11.7	15.0	0.84	0.069
	32	22.9	30.4	1.85	0.087	16.8	27.9	1.26	0.054	16.4	29.6	1.11	0.047	16.5	30.5	1.11	0.049
	64	29.3	55.3	2.32	0.069	21.3	45.7	1.65	0.038	19.8	54.2	1.45	0.035	20.3	56.2	1.44	0.036
ONETONE1	8	5.1	6.5	0.44	0.067	3.7	5.0	0.19	0.031	3.5	4.7	0.21	0.033	3.5	4.9	0.20	0.034
	16	8.7	11.6	0.62	0.051	7.8	10.2	0.34	0.026	7.6	9.6	0.38	0.032	7.8	10.1	0.36	0.029
	32	14.4	20.0	0.81	0.035	13.3	18.6	0.49	0.021	13.4	18.6	0.54	0.026	14.0	19.1	0.51	0.024
	64	19.9	30.2	1.08	0.024	19.9	31.5	0.65	0.017	19.6	30.5	0.72	0.018	19.3	30.4	0.69	0.019
LHR17	8	7.0	7.0	1.02	0.164	6.8	7.0	0.66	0.100	6.8	7.0	0.59	0.087	6.9	7.0	0.58	0.087
	16	14.4	15.0	1.40	0.117	12.2	15.0	0.91	0.074	12.3	15.0	0.81	0.064	12.3	15.0	0.80	0.063
	32	24.2	30.6	1.78	0.080	18.0	30.0	1.22	0.052	17.1	30.6	1.06	0.044	17.2	30.8	1.05	0.044
	64	31.4	53.3	2.21	0.062	22.9	51.9	1.58	0.037	20.7	55.0	1.37	0.031	20.8	55.8	1.36	0.032
LHR34	8	3.4	4.5	0.67	0.103	3.4	4.1	0.43	0.065	3.4	4.1	0.39	0.056	3.4	4.1	0.39	0.055
	16	7.3	8.6	1.02	0.086	7.1	8.4	0.66	0.053	7.2	8.3	0.59	0.046	7.1	8.3	0.59	0.046
	32	14.7	16.8	1.40	0.061	12.4	15.9	0.92	0.040	12.4	15.6	0.81	0.033	12.5	15.7	0.80	0.033
	64	24.2	31.4	1.78	0.043	18.2	30.3	1.22	0.028	17.3	30.8	1.06	0.023	17.3	31.0	1.06	0.023
BCSSTK32	8	3.6	5.3	0.07	0.016	3.1	4.6	0.05	0.013	3.9	5.8	0.06	0.014	3.4	5.2	0.05	0.012
	16	4.3	7.3	0.12	0.014	3.9	7.0	0.08	0.010	4.4	7.9	0.10	0.012	4.1	7.7	0.08	0.011
	32	5.1	9.5	0.19	0.011	4.4	8.9	0.14	0.008	4.7	9.9	0.15	0.009	4.6	9.4	0.14	0.009
	64	5.5	11.6	0.29	0.009	4.5	10.1	0.21	0.007	4.9	11.4	0.23	0.008	4.7	11.2	0.21	0.007
BCSSTK30	8	2.5	4.0	0.08	0.017	2.8	4.6	0.08	0.017	2.2	3.4	0.07	0.014	2.4	4.2	0.06	0.013
	16	3.6	6.2	0.18	0.018	3.4	6.0	0.14	0.015	3.0	5.0	0.14	0.016	3.1	5.2	0.13	0.014
	32	4.7	8.2	0.31	0.015	4.0	8.0	0.22	0.012	4.0	6.9	0.24	0.013	3.9	7.1	0.21	0.012
	64	5.7	10.0	0.50	0.013	4.6	9.0	0.34	0.010	4.5	8.4	0.37	0.010	4.5	9.3	0.34	0.010
Averages over K																	
	8	5.0	5.8	0.66	0.106	4.7	5.5	0.40	0.064	4.7	5.5	0.38	0.059	4.8	5.7	0.37	0.058
	16	9.4	11.2	0.94	0.079	8.5	10.9	0.59	0.048	8.4	10.8	0.55	0.045	8.6	11.1	0.54	0.044
	32	15.8	21.1	1.24	0.057	13.2	19.9	0.79	0.034	13.0	20.2	0.74	0.032	13.1	20.7	0.72	0.031
	64	22.0	36.5	1.57	0.042	17.3	30.8	1.03	0.024	16.7	32.5	0.96	0.022	16.5	33.6	0.94	0.023

In the "# of mssgs" column, "avg" and "max" denote the average and maximum number of messages, respectively, handled by a single processor. In the "comm. volume" column, "tot" denotes the total communication volume, whereas "max" denotes the maximum communication volume handled by a single processor. Communication volume values (in terms of the number of words transmitted) are scaled by the number of rows/columns of the respective test matrices.

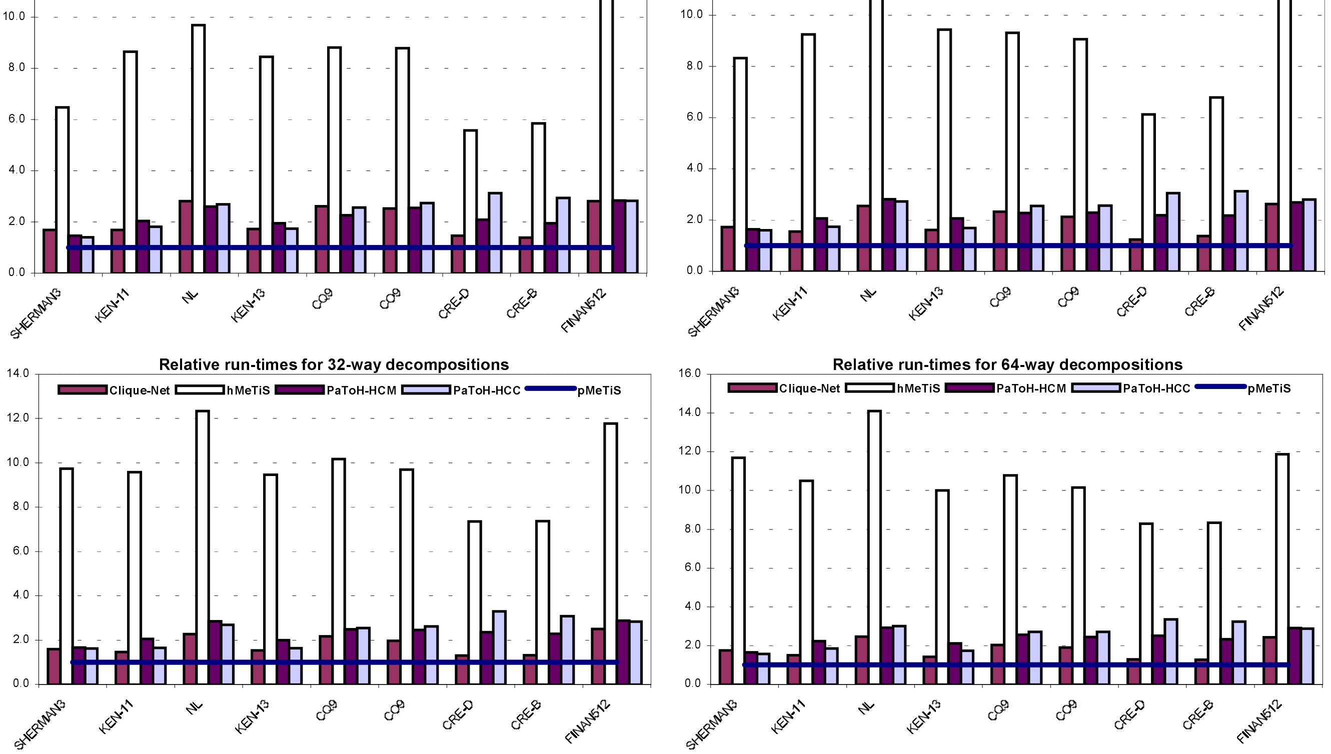


Figure 7: Relative run-time performance of the proposed column-net/row-net hypergraph model (Clique-net, hMeTiS, PaToH-HCM and PaToH-HCC) to the graph model (pMeTiS) in rowwise/columnwise decomposition of symmetric test matrices. Bars above 1.0 indicate that the hypergraph model leads to slower decomposition time than the graph model.

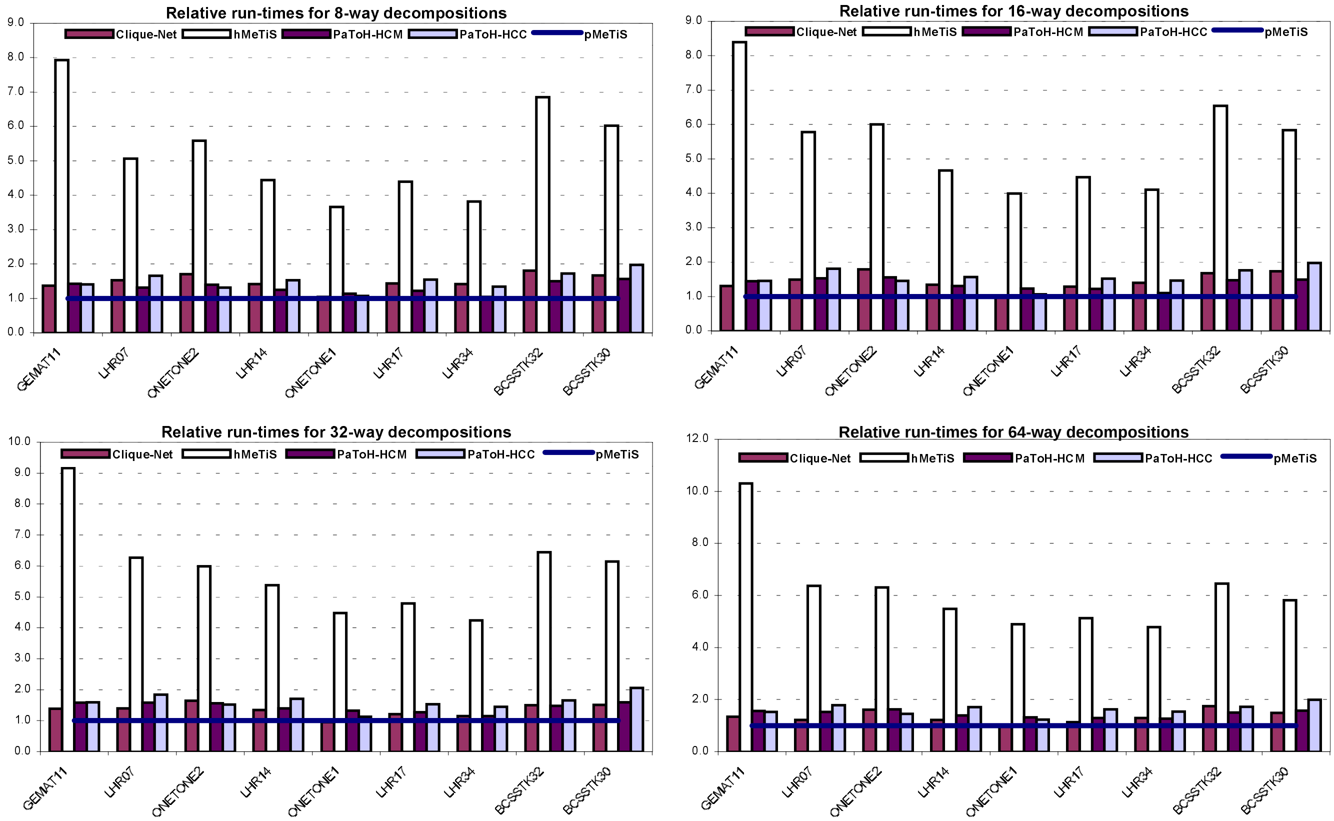


Figure 8: Relative run-time performance of the proposed column-net hypergraph model (Clique-net, hMeTiS, PaToH-HCM and PaToH-HCC) to the graph model (pMeTiS) in rowwise decomposition of symmetric test matrices. Bars above 1.0 indicate that the hypergraph model leads to slower decomposition time than the graph model.

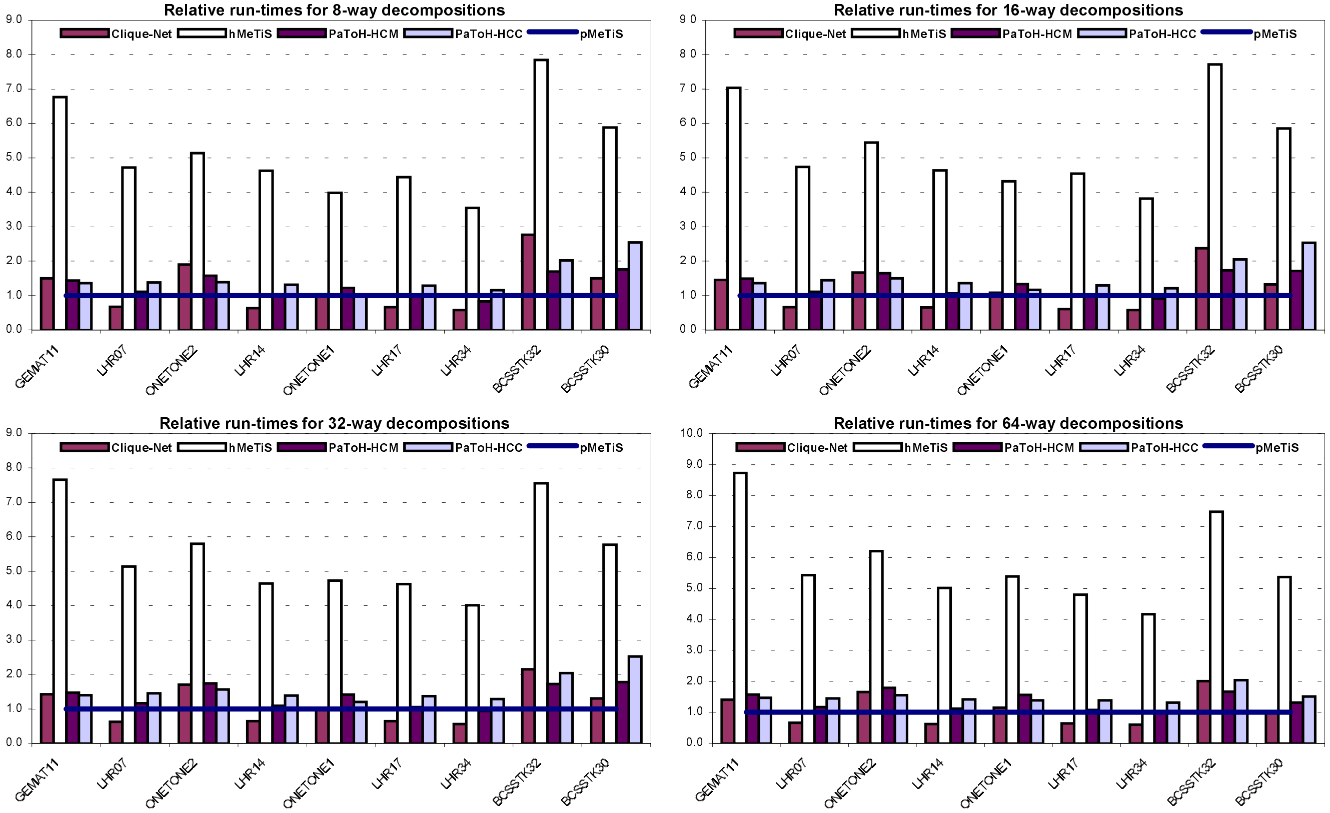


Figure 9: Relative run-time performance of the proposed row-net hypergraph model (Clique-net, hMeTiS, PaToH-HCM and PaToH-HCC) to the graph model (pMeTiS) in columnwise decomposition of symmetric test matrices. Bars above 1.0 indicate that the hypergraph model leads to slower decomposition time than the graph model.

Table V: Overall performance averages of the proposed hypergraph models normalized with respect to those of the graph models using pMeTiS.

K	pMeTiS (clique-net model)				hMeTiS				PaToH-HCM				PaToH-HCC			
	Tot. Comm. Volume			Time	Tot. Comm. Volume			Time	Tot. Comm. Volume			Time	Tot. Comm. Volume			Time
	best	worst	avg		best	worst	avg		best	worst	avg		best	worst	avg	
Symmetric Matrices: Column-net Model ≡ Row-net Model																
8	0.86	0.84	0.85	2.08	0.73	0.70	0.71	8.13	0.73	0.73	0.73	2.19	0.73	0.73	0.73	2.42
16	0.86	0.84	0.83	1.90	0.70	0.66	0.66	8.95	0.70	0.69	0.68	2.25	0.71	0.69	0.69	2.43
32	0.85	0.84	0.84	1.79	0.68	0.65	0.66	9.72	0.69	0.68	0.68	2.33	0.69	0.68	0.68	2.44
64	0.85	0.84	0.84	1.78	0.71	0.68	0.69	10.64	0.72	0.69	0.70	2.41	0.72	0.69	0.70	2.56
avg	0.86	0.84	0.84	1.89	0.70	0.67	0.68	9.36	0.71	0.70	0.70	2.30	0.71	0.70	0.70	2.46
Nonsymmetric Matrices: Column-net Model																
8	0.78	0.78	0.78	1.48	0.68	0.63	0.64	5.31	0.67	0.64	0.64	1.32	0.66	0.62	0.63	1.50
16	0.80	0.78	0.78	1.44	0.66	0.63	0.64	5.53	0.67	0.64	0.65	1.37	0.65	0.62	0.63	1.56
32	0.79	0.78	0.78	1.34	0.66	0.64	0.66	5.88	0.67	0.65	0.66	1.44	0.65	0.63	0.64	1.61
64	0.80	0.79	0.79	1.34	0.69	0.68	0.68	6.17	0.69	0.68	0.68	1.45	0.67	0.66	0.66	1.62
avg	0.79	0.78	0.79	1.40	0.67	0.64	0.66	5.72	0.67	0.65	0.66	1.39	0.66	0.63	0.64	1.57
Nonsymmetric Matrices: Row-net Model																
8	0.75	0.74	0.76	1.25	0.64	0.62	0.63	5.22	0.64	0.63	0.63	1.29	0.62	0.60	0.61	1.50
16	0.75	0.74	0.75	1.15	0.65	0.63	0.64	5.34	0.65	0.63	0.65	1.33	0.62	0.61	0.62	1.54
32	0.75	0.75	0.75	1.12	0.67	0.65	0.66	5.55	0.66	0.64	0.66	1.38	0.63	0.62	0.63	1.58
64	0.76	0.77	0.76	1.09	0.67	0.67	0.67	5.84	0.66	0.65	0.66	1.36	0.64	0.63	0.63	1.50
avg	0.75	0.75	0.76	1.15	0.66	0.64	0.65	5.49	0.65	0.64	0.65	1.34	0.63	0.61	0.62	1.53

In total communication volume, a ratio smaller than 1.00 indicates that the hypergraph model produces better decompositions than the graph model. In execution time, a ratio greater than 1.00 indicates that the hypergraph model leads to slower decomposition time than the graph model.

Ümit V. Çatalyürek received the B.S. and M.S. degrees in computer engineering and information science from Bilkent University, Ankara, Turkey, in 1992 and 1994, respectively. He is currently working towards the Ph.D. degree in the Department of Computer Engineering and Information Science, Bilkent University, Ankara, Turkey. His current research interests are parallel computing and graph/hypergraph partitioning.

Cevdet Aykanat received the B.S and M.S. degrees from Middle East Technical University, Ankara, Turkey, in 1977 and 1980, respectively, and the Ph.D. degree from Ohio State University, Columbus, in 1988, all in electrical engineering. He was a Fulbright scholar during his Ph.D. studies. He worked at the Intel Supercomputer Systems Division, Beaverton, OR, as a research associate. Since October 1988 he has been with the Department of Computer Engineering and Information Science, Bilkent University, Ankara, Turkey, where he is currently an associate professor. His research interests include parallel computer architectures, parallel algorithms, applied parallel computing, neural network algorithms and graph/hypergraph partitioning. He is a member of the ACM, IEEE and IEEE Computer Society.