PERMUTING SPARSE RECTANGULAR MATRICES INTO BLOCK-DIAGONAL FORM

CEVDET AYKANAT*, ALI PINAR†, AND ÜMIT V. ÇATALYÜREK‡

Abstract. This work investigates the problem of permuting a sparse rectangular matrix into block diagonal form. Block diagonal form of a matrix grants an inherent parallelism for the solution of the deriving problem, which has been recently investigated in the context of mathematical programming, LU factorization and QR factorization. We propose bipartite graph and hypergraph models to represent the nonzero structure of a matrix, which reduce the permutation problem to those of graph partitioning by vertex separator and hypergraph partitioning, respectively. Besides proposing the models to represent sparse matrices and investigating related combinatorial problems, we provide a detailed survey of relevant literature to bridge the gap between different societies, investigate existing techniques for partitioning and propose new ones, and finally present a thorough empirical study of these techniques. Our experiments on a wide range of matrices, using state-of-the-art graph and hypergraph partitioning tools MeTiS and PaToH, revealed that the proposed methods yield very effective solutions both in terms of solution quality and runtime.

Key words. Block-angular LP problems, coarse-grain parallelism, sparse rectangular matrices, singly-bordered block-diagonal form, doubly-bordered block-diagonal form, graph partitioning by vertex separator, hypergraph partitioning.

AMS subject classifications. 65F05, 65F50, 65F20,65K05, 65Y05,05C50, 05C65, 05C85, 05C90

1. Introduction.

Solution methods for various applications (e.g., linear programming (LP) problems, systems of linear equations, and least squares problems) exploit the block-diagonal structure of a sparse matrix for coarse-grain parallelization. In these methods, block diagonals give rise to subproblems that can be solved independently and concurrently, whereas the border incurs a coordination task to combine the subproblem solutions into a solution of the original problem, which is usually not amenable to efficient parallelization. The objective of this work is to enhance these decomposition-based solution methods by transforming the underlying matrix into a block-diagonal form with small border size while maintaining a given balance condition on the sizes of the diagonal blocks.

The target problem can be formally described as permuting rows and columns of an $M \times N$ sparse matrix A into a K-way singly-bordered block-diagonal (SB) form:

$$A^{\pi} = P\,A\,Q = \begin{bmatrix} A^{\pi}_{11} & \dots & A^{\pi}_{1K} \\ \vdots & \ddots & \vdots \\ A^{\pi}_{K1} & \dots & A^{\pi}_{KK} \\ A^{\pi}_{S1} & \dots & A^{\pi}_{SK} \end{bmatrix} = \begin{bmatrix} B_1 & & \\ & \ddots & \\ & & B_K \\ R_1 & \dots & R_K \end{bmatrix} = A_{SB} \qquad (1.1)$$

where P and Q denote, respectively, the row and column permutation matrices to be determined. In (1.1), $A^{\pi}_{kk} = B_k$ is an $M_k \times N_k$ submatrix and $A^{\pi}_{Sk} = R_k$ is an $M_c \times N_k$ submatrix, for $k = 1, 2, \dots, K$, where $M = M_c + \sum_{k=1}^{K} M_k$ and $N = \sum_{k=1}^{K} N_k$. Each row of the $M_c \times N$ border submatrix $R = (R_1 \;\cdots\; R_K)$ is called a *column-coupling* or simply *coupling* row. Each coupling row has nonzeros in the columns of

*Computer Engineering Department, Bilkent University, Ankara, Turkey (aykanat@cs.bilkent.edu.tr)

†NERSC, Lawrence Berkeley Laboratory (apinar@lbl.gov)

‡Department of Biomedical Informatics, The Ohio State Univesity, Columbus, 43210, OH (catalyurek.1@osu.edu)

at least two diagonal blocks. Hence, the objective is to permute matrix A into an SB form A_{SB} such that the number (M_c) of coupling rows is minimized while a given balance criterion on the row and column dimensions of the diagonal block matrices is maintained. The SB form in (1.1) is referred to here as the *primal* SB form, whereas in the *dual* SB form it is the columns that constitute the border.

We also consider the problem of permuting rows and columns of an $M\times N$ sparse matrix A into a K-way doubly-bordered block-diagonal (DB) form:

$$(1.2)\ A^{\pi}=PAQ=\begin{bmatrix} A^{\pi}_{11} & \cdots & A^{\pi}_{1K} & A^{\pi}_{1S} \\ \vdots & \ddots & \vdots & \vdots \\ A^{\pi}_{K1} & \cdots & A^{\pi}_{KK} & A^{\pi}_{KS} \\ A^{\pi}_{S1} & \cdots & A^{\pi}_{SK} & A^{\pi}_{SS} \end{bmatrix} = \begin{bmatrix} B_1 & & & C_1 \\ & \ddots & & \\ & & B_K & C_K \\ R_1 & \cdots & R_K & D \end{bmatrix} = A_{DB}$$

In (1.2), $A^{\pi}_{kk}=B_k$ is an $M_k\times N_k$ diagonal submatrix, $A^{\pi}_{Sk}=R_k$ is an $M_c\times N_k$ submatrix, and $A^{\pi}_{kS}=C_k$ is an $M_k\times N_c$ submatrix for $k=1,2,\ldots,K$, where $M=M_c+\sum_{k=1}^{K} M_k$ and $N=N_c+\sum_{k=1}^{K} N_k$. Each row and column of matrix $R=(R_1\ \cdots\ R_K\ D)$ and $C=(C_1^T\ \cdots\ C_K^T\ D^T)^T$ is called a coupling row and a coupling column, respectively. The objective is to permute matrix A into a DB form A_{DB} such that the number of coupling rows and columns is minimized while a given balance criterion on the row and column dimensions of the diagonal block matrices is maintained.

The literature that addresses this problem is very rare and recent. Ferris and Horn [14] proposed a two-phase approach for A-to-A_{SB} transformation. In the first phase, matrix A is transformed into a DB form A_{DB} as an intermediate form. In the second phase, A_{DB} is transformed into an SB form through *column-splitting* as discussed in Section 3.3. Our initial results of this problem were presented in two conference papers [55, 56]. In [55], we proposed the basics of our hypergraph model and how to exploit this model to permute matrices to block-diagonal form. In our subsequent work [56] we proposed our graph models. Later Hu et. al [32] independently investigated the same problem without spelling out the exact model to represent the sparsity structures of matrices or the details of their algorithm for permutation. In this paper, we present a complete work on the problem of permuting sparse matrices to block-diagonal form. We consider permutations to DB form as well as permutations to primal and dual SB form, provide a detailed survey of relevant literature to bridge the gap between different societies, investigate existing techniques for partitioning and propose new ones, and finally present a thorough empirical study of these techniques.

Our proposed graph and hypergraph models for sparse matrices reduce the problem of permuting a sparse matrix to block-diagonal form to the well–known problems of graph partitioning by vertex separator (GPVS) and hypergraph partitioning (HP). GPVS is widely used in nested-dissection based low-fill orderings for factorization of symmetric, sparse matrices, whereas HP is widely used for solving the circuit partitioning and placement problems in VLSI layout design. Our models enable the use of algorithms and tools for these well–studied problems to permute sparse matrices to block-diagonal form efficiently and effectively. Our contributions, however, are not restricted to application of standard techniques to this problem, but we also propose techniques to improve the state of the art in GPVS and HP, as well as techniques that exploit the special structure of this problem.

In Section 4, we show that the A-toA_{DB} transformation problem can be described as a GPVS problem on the bipartite graph representation of A. The objective in the K-way GPVS problem is to find a subset of vertices (vertex separator) of minimum size that disconnects the K vertex parts, while maintaining a given balance criterion on the vertex counts of K parts. In this model, minimizing the size of the vertex separator corresponds to minimizing the sum of the number of coupling rows and linking columns in A_{DB}.

There are two heuristic approaches, referred to here as direct and indirect approaches, for solving the GPVS problem. The direct GPVS approach directly aims at a vertex separator on the graph, whereas the indirect approach finds a vertex separator through graph partitioning by edge separator (GPES). After finding an edge separator, the indirect approach takes the boundary vertices as the wide separator to be refined to a narrow separator, hoping that a small edge separator yields a small vertex separator. The approach adopted by Ferris and Horn [14] falls into this class.

In Section 5, we propose two greedy heuristics for indirect GPVS. The first one is a new edge weighting scheme on the bipartite graph so that minimizing the weighted edge cut through a GPES tool is expected to produce a wide vertex separator with "better" quality for refinement. The second one is a heuristic for finding a good vertex cover of a K-partite graph. This heuristic is used to find a wide–to–narrow vertex–separator refinement from the K-way partition produced by the GPES tool. The vertex-cover model has been widely used for separator refinement in nested-dissection based symmetric matrix ordering. In Section 5, we also show the flaws of the vertex-cover model in separator refinement.

In Section 6, we propose a one-phase approach for permuting A directly into an SB form. In this approach, a hypergraph model—proposed in an earlier version of this work [55]—is exploited to represent rectangular matrices. The proposed model reduces the A-to-A_{SB} transformation problem into the HP problem. In this model, minimizing the size of the hyperedge separator directly corresponds to minimizing the number of coupling rows in A_{SB}.

In Section 7.1, we briefly discuss alternative circuit representation models proposed by the VLSI community for circuit partitioning and placement. These alternative models can also be considered as efforts towards using GP algorithms and tools to solve the HP problem. However, the GPES-based HP models minimize the wrong metric, that is minimizing the cutsize in GPES is only an approximation to minimizing the cutsize in HP. In Section 7.2, we propose a GPVS-based model for solving the HP problem through the *net intersection graph* representation of a given hypergraph. The proposed model achieves a one-to-one correspondence between the cutsizes of GPVS and HP. In Section 7.3, we display the matrix theoretical view of the proposed model and show the close relation between the problems of permuting matrix A into an SB form and permuting matrix AA^T into a DB form. In Section 8, we present a brief overview on the state-of-the-art graph and hypergraph partitioning algorithms and tools. Finally in Section 9, we test the performance of our proposed models and associated solution approaches on a wide range of large LP constraint matrices.

2. Applications.

Block diagonal structure of a matrix grants an inherent parallelism for the solution of the deriving problem. In this section, we will exemplify how to exploit this parallelism in three fundamental problems of linear algebra and optimization: linear programming, linear equations, and least-squares.

Linear Programming: Exploiting the block-angular structure of linear programs (LPs) dates back to the work of Dantzig [13], when the motivation was solving large LPs with limited memory. Later works investigated parallelization techniques [20, 31, 50]. The proposed techniques [13, 47, 51, 60] lead to iterative algorithms, where each iteration involves solving K independent LP subproblems corresponding to the block constraints followed by a coordination phase for coordinating the solutions of the subproblems according to the coupling constraints. These approaches have two nice properties. First, as the solution times of most LP's in practice increase as a quadratic or cubic function with the size of the problem, it is more efficient to solve a set of small problems than a single aggregate problem. Second, they give rise to a natural, coarse-grain parallelism that can be exploited by processing the subproblems concurrently. Coarse-grain parallelism inherent in these approaches has been exploited in several successful parallel implementations on distributed-memory multicomputers through manager-worker scheme [14, 20, 31, 50]. At each iteration, the LP subproblems are solved concurrently by worker processors, whereas a serial master problem is solved by the manager processor in the coordination phase.

As recently proposed [14], these successful decomposition-based approaches can be exploited for coarse-grain parallel solution of general LP problems by transforming them into block-angular forms. In matrix theoretical view, this transformation problem can be described as permuting the rectangular constraint matrix of the LP problem into an SB form as shown in (1.1) with minimum border size while maintaining a given balance criterion on the diagonal blocks. Note that row and column permutation correspond to reordering of the constraints and variables of the given LP problem. Here, minimizing the border size relates to minimizing the size of the master problem. The size of the master problem has been reported to be crucial for the parallel performance of these algorithms [14, 50]. First, it affects the convergence of the overall iterative algorithm. Second, in most algorithms the master problem is solved serially by the manager processor. Finally, it determines the communication requirement between phases. It is also important to have equal-sized blocks for load balance for the efficiency of the parallel phase.

It is worth noting that exploiting the block angular structure of the constraint matrices is not restricted to linear programs and can be applied in different optimization problems [52, 61].

Linear Equations: In most scientific computing applications, the core of the computation is solving a system of linear equations. Direct methods like LU factorization are commonly used for the solution of nonsymmetric systems for their numerical robustness. A coarse-grain parallel LU factorization scheme [32, 59] is to permute the square, nonsymmetric coefficient matrix to a DB form as shown in (1.2). Notice that diagonal blocks of the permuted matrix constitute independent subproblems, and can be factored concurrently. Pivots are chosen within the blocks for concurrency. Rows/columns that cannot be eliminated including those that cannot be eliminated due to numerical reasons are permuted to the end of the matrix to achieve a partially-factored matrix in DB form as

$$\begin{bmatrix} L_1U_1 & & & U'_1 \\ & \ddots & & \vdots \\ & & L_KU_K & U'_K \\ L'_1 & \dots & L'_K & F \end{bmatrix}$$

In this matrix, L_kU_k constitutes the factored form of $A^{\pi}_{k} = B_k$ after the unfactored rows/columns are permuted to the end of the matrix. In a subsequent phase, the coupling rows and columns, and unfactored columns and rows from the blocks are factored. It is possible to parallelize this step with different (and usually less efficient) techniques.

We stated two objectives during permutation to DB form. First one is minimizing the number of coupling rows and columns, which relates to minimizing the work for the second phase, thus increasing concurrency. Our second objective of equal-sized blocks provides load balance during factorization of the blocks.

Least Squares Problems: Least squares is one of the fundamental problems in numerical linear algebra defined as follows:

$$\min_{\mathrm{x}} \| A\mathrm{x} - \mathrm{b} \|_2,$$

where A is an $M \times N$ matrix with $M \geq N$. QR factorization is a method commonly used to solve least-squares problems. In this method, matrix A is factored into an orthogonal $M \times M$ matrix Q and an upper triangular $N \times N$ matrix R with nonnegative diagonal elements so that

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix}$$

Then, we can solve for $R\mathrm{x} = \mathrm{b}'$ to get a solution, where b' is composed of the first N entries of vector b.

Computationally, this problem is very similar to LU factorization, thus we can use the same scheme to parallelize QR factorization. Given a matrix in dual SB form,

$$\begin{bmatrix} B_1 & & & C_1 \\ & \ddots & & \vdots \\ & & B_K & C_K \end{bmatrix}$$

the diagonal blocks of the matrix constitute the independent subblocks and can be factored independently. Thus first phase is composed of factoring B_k and the associated coupling columns in C_k concurrently, so that

$$[B_k \quad C_k] = Q_k \begin{bmatrix} R_k & S_k \\ 0 & C'_k \end{bmatrix} \quad \text{for } k = 1, 2, \ldots, K.$$

In a subsequent phase, we factor $C' = \left[\; C'_1, \ldots, C'_K \; \right]^T$ [4].

So, in permuting a given matrix A into a dual SB form, minimizing the number of coupling columns minimizes the work on the second phase of the algorithm, and equal-sized blocks provide load balance for the first phase.

3. Preliminaries.

3.1. Graph Partitioning. 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. We use the notation $Adj(v_i)$ to denote the set of vertices adjacent to vertex v_i in graph $\mathcal{G}$. We extend this operator to include the adjacency set of a vertex subset $\mathcal{V}' \subset \mathcal{V}$, i.e., $Adj(\mathcal{V}') = \{v_j \in \mathcal{V} - \mathcal{V}' : v_j \in Adj(v_i) \text{ for some } v_i \in \mathcal{V}'\}$. The degree d_i of a vertex v_i is equal to the number of edges incident to v_i, i.e., $d_i = |Adj(v_i)|$. An edge subset $\mathcal{E}_S$ is a K-way *edge separator* if its removal disconnects the graph into at least K connected components. A vertex subset $\mathcal{V}_S$ is a K-way *vertex separator* if the subgraph induced by the vertices in $\mathcal{V} - \mathcal{V}_S$ has at least K connected components.

$\Pi_{GPES} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K\}$ is a K-way vertex partition of $\mathcal{G}$ by edge separator $\mathcal{E}_S \subset \mathcal{E}$ if the following conditions hold: $\mathcal{V}_k \subset \mathcal{V}$ and $\mathcal{V}_k \neq \emptyset$ for $1 \leq k \leq K$; $\mathcal{V}_k \cap \mathcal{V}_\ell = \emptyset$

for $1 \le k < \ell \le K$; $\bigcup_{k=1}^{K} \mathcal{V}_k = \mathcal{V}$. Note that all edges between the vertices of different parts belong to $\mathcal{E}_S$. Edges in $\mathcal{E}_S$ are called *cut (external)* edges and all other edges are called *uncut (internal)* edges. In a partition Π_{GPES} of $\mathcal{G}$, a vertex is said to be a *boundary* vertex if it is incident to a cut edge. The *cutsize* definition for representing the cost of a partition Π_{GPES} is

$$cost(\Pi_{GPES}) = \sum_{e_{ij} \in \mathcal{E}_S} w_{ij}, \tag{3.1}$$

where each cut edge $e_{ij} = (v_i, v_j)$ contributes its weight w_{ij} to the cost. Hence, the K-way GPES problem can be defined as the task of dividing a graph into K parts such that the cutsize is minimized, while a given balance criterion on part sizes is maintained.

$\Pi_{GPVS} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K; \mathcal{V}_S\}$ is a K-way vertex partition of $\mathcal{G}$ by vertex separator $\mathcal{V}_S \subset \mathcal{V}$ if the following conditions hold: $\mathcal{V}_k \subset \mathcal{V}$ and $\mathcal{V}_k \neq \emptyset$ for $1 \le k \le K$; $\mathcal{V}_k \cap \mathcal{V}_\ell = \emptyset$ for $1 \le k < \ell \le K$ and $\mathcal{V}_k \cap \mathcal{V}_S = \emptyset$ for $1 \le k < K$; $\bigcup_{k=1}^{K} \mathcal{V}_k \cup \mathcal{V}_S = \mathcal{V}$; the removal of $\mathcal{V}_S$ gives K disconnected parts $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K$ (i.e., $Adj(\mathcal{V}_k) \subseteq \mathcal{V}_S$ for $1 \le k \le K$). A vertex $v_i \in \mathcal{V}_k$ is said to be a boundary vertex of part $\mathcal{V}_k$ if it is adjacent to a vertex in $\mathcal{V}_S$. A vertex separator is said to be *narrow* if no subset of it forms a separator, and *wide* otherwise. The cost of a partition Π_{GPVS} is

$$cost(\Pi_{GPVS}) = |\mathcal{V}_S|\,. \tag{3.2}$$

Hence, the K-way GPVS problem can be defined as the task of finding a K-way vertex separator of minimum size, while maintaining a given balance criterion on the sizes of the K parts. Both GPES and GPVS problems are known to be NP-hard [6].

As mentioned earlier, indirect GPVS approaches first perform a GPES on the given graph to minimize the number of cut edges (i.e., $w_{ij} = 1$ in (3.1)) and then take the boundary vertices as the wide separator to be refined to a narrow separator. The wide-to-narrow refinement problem is described as a *minimum vertex cover* problem on the subgraph induced by the cut edges [58]. A minimum vertex cover is taken as a narrow separator for the whole graph, since each cut edge will be adjacent to a vertex in the vertex cover. That is, let $\mathcal{V}_{Bk} \subseteq \mathcal{V}_k$ denote the set of boundary vertices of part $\mathcal{V}_k$ in a partition $\Pi_{GPES} = \{\mathcal{V}_1, \ldots, \mathcal{V}_K\}$ of a given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ by edge separator $\mathcal{E}_S \subset \mathcal{E}$. Then, $\mathcal{K}(\mathcal{E}_S) = (\mathcal{V}_B = \cup_{k=1}^{K} \mathcal{V}_{Bk}, \mathcal{E}_S)$ denotes the K-partite subgraph of $\mathcal{G}$ induced by $\mathcal{E}_S$. A vertex cover $\mathcal{V}_S = \cup_{k=1}^{K} \mathcal{V}_{Sk}$ on $\mathcal{K}(\mathcal{E}_S)$ constitutes a K-way GPVS $\Pi_{GPVS} = \{\mathcal{V}_1 - \mathcal{V}_{S1}, \ldots, \mathcal{V}_K - \mathcal{V}_{SK}; \mathcal{V}_S\}$ of $\mathcal{G}$, where $\mathcal{V}_{Sk} \subseteq \mathcal{V}_{Bk}$ denotes the subset of boundary vertices of part $\mathcal{V}_k$ that belong to the vertex cover of $\mathcal{K}(\mathcal{E}_S)$. A minimum vertex cover $\mathcal{V}_S$ of $\mathcal{K}(\mathcal{E}_S)$ corresponds to an optimal refinement of the wide separator $\mathcal{V}_B$ into a narrow separator $\mathcal{V}_S$ under the assumption that each boundary vertex is adjacent to at least one non-boundary vertex in Π_{GPES} (see Section 5.3).

3.2. Hypergraph Partitioning. A hypergraph $\mathcal{H} = (\mathcal{U}, \mathcal{N})$ is defined as a set of nodes (vertices) $\mathcal{U}$ and a set of nets (hyperedges) $\mathcal{N}$ among those vertices. We refer to the vertices of $\mathcal{H}$ as nodes to avoid the confusion between graphs and hypergraphs. Every net $n_i \in \mathcal{N}$ is a subset of nodes, i.e., $n_i \subseteq \mathcal{U}$. The nodes in a net n_i are called its *pins* and denoted as $Pins(n_i)$. We extend this operator to include the pin list of a net subset $\mathcal{N}' \subset \mathcal{N}$, i.e., $Pins(\mathcal{N}') = \bigcup_{n_i \in \mathcal{N}'} Pins(n_i)$. The size s_i of a net n_i is equal to the number of its pins, i.e., $s_i = |Pins(n_i)|$. The set of nets connected to a node u_j is denoted as $Nets(u_j)$. We also extend this operator to include the net list of a node subset $\mathcal{U}' \subset \mathcal{U}$, i.e., $Nets(\mathcal{U}') = \bigcup_{u_j \in \mathcal{U}'} Nets(u_j)$. The degree d_j of a node u_j is equal to the number of nets it is connected to, i.e., $d_j = |Nets(u_j)|$. The total number p of pins denote the size of $\mathcal{H}$ where $p = \sum_{n_i \in \mathcal{N}} s_i = \sum_{u_j \in \mathcal{U}} d_j$. Graph is a special instance of hypergraph such that each net has exactly two pins.

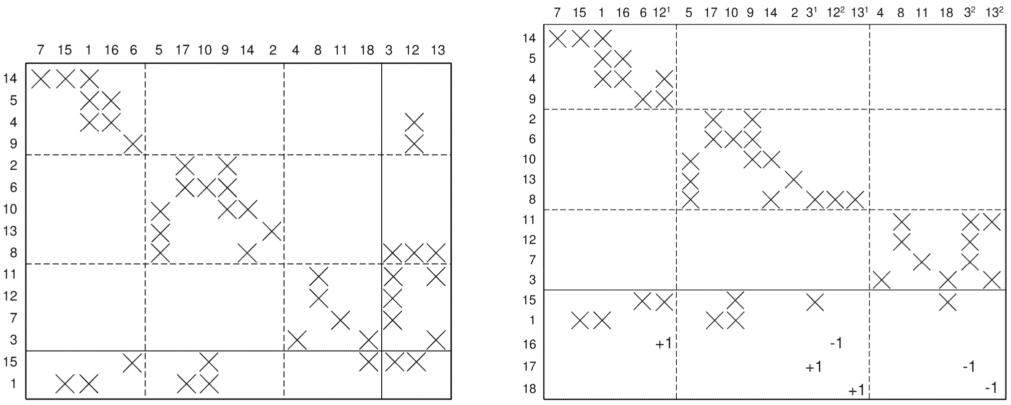


FIG. 3.1. *Column-splitting process.*

$\Pi_{HP}=\{\mathcal{U}_1,\mathcal{U}_2,\ldots,\mathcal{U}_K\}$ is a K-way node partition of $\mathcal{H}$ if the following conditions hold: $\mathcal{U}_k\subset\mathcal{U}$ and $\mathcal{U}_k\neq\emptyset$ for $1\leq k\leq K$; $\mathcal{U}_k\cap\mathcal{U}_\ell=\emptyset$ for $1\leq k<\ell\leq K$; $\bigcup_{k=1}^{K}\mathcal{U}_k=\mathcal{U}$. In a partition Π_{HP} of $\mathcal{H}$, a net that has at least one pin (node) in a part is said to *connect* that part. *Connectivity set* Λ_i of a net n_i is defined as the set of parts connected by n_i. *Connectivity* $\lambda_i=|\Lambda_i|$ of a net n_i denotes the number of parts connected by n_i. A net n_i is said to be *cut (external)* if it connects more than one part (i.e., $\lambda_i>1$), and *uncut (internal)* otherwise (i.e., $\lambda_j=1$). A net n_i is said to be an internal net of a part $\mathcal{U}_k$ if it connects only part $\mathcal{U}_k$, i.e., $\Lambda_i=\{\mathcal{U}_k\}$ which also means $Pins(n_i)\subseteq\mathcal{U}_k$. The set of internal nets of a part $\mathcal{U}_k$ is denoted as $\mathcal{N}_k$, for $k=1,\ldots,K$, and the set of external nets of a partition Π_{HP} is denoted as $\mathcal{N}_S$. So, although Π_{HP} is defined as a K-way partition on the node set of $\mathcal{H}$, it can also be considered as inducing a $(K+1)$-way partition $\{\mathcal{N}_1,\ldots,\mathcal{N}_K;\mathcal{N}_S\}$ on the net set. $\mathcal{N}_S$ can be considered as a net separator whose removal gives K disconnected node parts $\mathcal{U}_1,\ldots,\mathcal{U}_K$ as well as K disconnected net parts $\mathcal{N}_1,\ldots,\mathcal{N}_K$. Two cutsize definitions widely used in VLSI community [48] for representing the cost of a partition Π_{HP} are:

$$(3.3)\quad (a)\quad cost(\Pi_{HP})=|\mathcal{N}_S|\qquad\text{and}\qquad(b)\quad cost(\Pi_{HP})=\sum_{n_i\in\mathcal{N}_S}(\lambda_i-1).$$

Hence, the K-way HP problem can be defined as the task of dividing a hypergraph into K parts such that the cutsize is minimized, while a given balance criterion among part sizes is maintained. The HP problem is known to be NP-hard [48].

3.3. Column-Splitting Method for A_{DB}-to-A_{SB} Transformation. In the second phase of FH algorithm [14], A_{DB} is transformed into an SB form through the *column-splitting* technique used in stochastic programming to treat anticipativity [53]. In this technique, we consider the variables corresponding to the linking columns. Consider a linking column c_j in submatrix $C=(C_1^T\ \cdots\ C_k^T\ \cdots\ C_K^T\ D^T)^T$ of A_{DB}, and let Λ_j denote the set of C_k's that have at least one nonzero in column c_j. The nonzeros of a linking column c_j is split into $|\Lambda_j|-1$ columns such that each new column includes nonzeros in rows of only one block. That is, we introduce one copy c_j^k of column c_j for each block $C_k\in\Lambda_j$ to decouple C_k from all other blocks in Λ_j on variable x_j, so that c_j^k is permuted to be a column of B_k. Then, we add $|\Lambda_j|$–1 coupling constraints as coupling rows into A_{DB} that force these variables $\{x_j^k : C_k\in\Lambda_j\}$ all to be equal. Note that this splitting process for column c_j increases both the row and column dimensions of matrix A_{SB} by $|\Lambda_j|-1$. Fig. 3.1 depicts the column-splitting process on the A_{DB} matrix obtained in Fig. 4.2b.

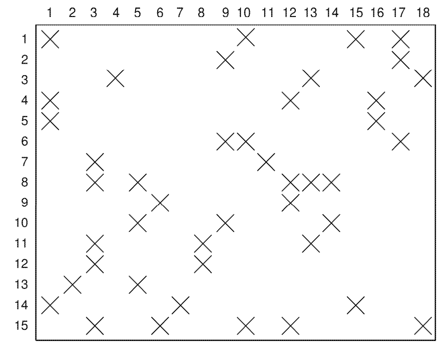


FIG. 4.1. *A 15×18 sample matrix A.*

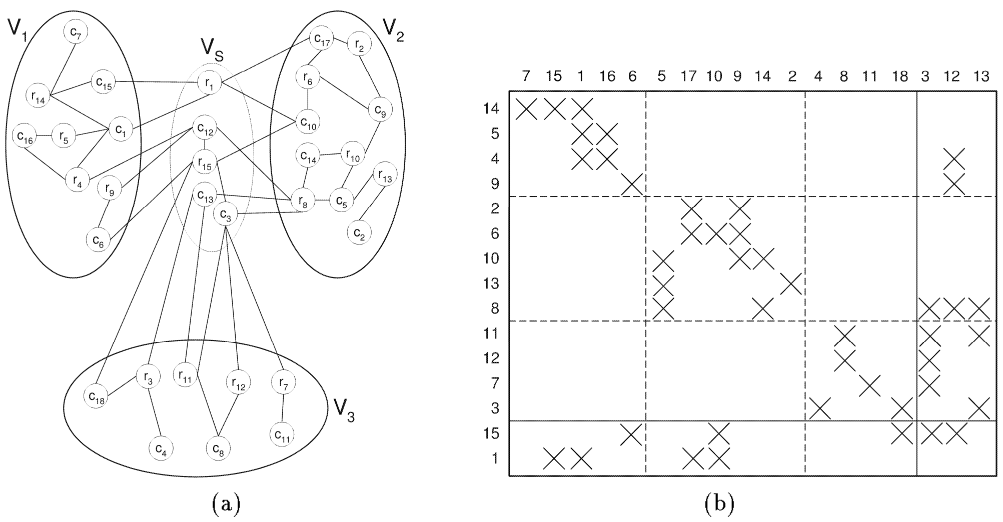


FIG. 4.2. *(a) Bipartite graph representation $\mathcal{B}_A$ of the sample A matrix given in Fig. 4.1 and 3-way partitioning Π_{GPVS} of $\mathcal{B}_A$ by vertex separator, (b) 3-way DB form A_{DB} of A induced by Π_{GPVS}.*

4. Bipartite Graph Model for A-to-A_{DB} Transformation. In this section, we show that the A-to-A_{DB} transformation problem can be described as a GPVS problem on the bipartite graph representation of A. In the bipartite graph model, $M\times N$ matrix $A=(a_{ij})$ is represented as a bipartite graph $\mathcal{B}_A=(\mathcal{V}=\mathcal{R}\cup\mathcal{C},\mathcal{E})$ on $M+N$ vertices with the number of edges equal to the number of nonzeros in A. Each row and column of A is represented by a vertex in $\mathcal{B}_A$ so that vertex sets $\mathcal{R}$ and $\mathcal{C}$ representing the rows and columns of A, respectively, form the vertex bipartition $\mathcal{V}=\mathcal{R}\cup\mathcal{C}$ with $|\mathcal{R}|=M$ and $|\mathcal{C}|=N$. There exists an edge between a row vertex $r_i\in\mathcal{R}$ and a column vertex $c_j\in\mathcal{C}$ if and only if the respective matrix entry a_{ij} is nonzero. So, $Adj(r_i)$ and $Adj(c_j)$ effectively represent the sets of columns and rows that have nonzeros in row i and column j, respectively. Fig. 4.2a displays the bipartite graph representation of the sample matrix given in Fig. 4.1.

Consider a K-way partition $\Pi_{GPVS}=\{\mathcal{V}_1,\ldots,\mathcal{V}_K;\mathcal{V}_S\}$ of $\mathcal{B}_A$, where $\mathcal{V}_k=\mathcal{R}_k\cup\mathcal{C}_k$ for $k=1,\ldots,K$ and $\mathcal{V}_S=\mathcal{R}_S\cup\mathcal{C}_S$ with $\mathcal{R}_k,\mathcal{R}_S\subseteq\mathcal{R}$ and $\mathcal{C}_k,\mathcal{C}_S\subseteq\mathcal{C}$. Π_{GPVS} can be decoded as a partial permutation on the rows and columns of A to induce a permuted matrix A^π. In this permutation, the rows and columns associated with the vertices in $\mathcal{R}_{k+1}$ and $\mathcal{C}_{k+1}$ are ordered after the rows and columns associated with the vertices in $\mathcal{R}_k$ and $\mathcal{C}_k$, for $k=1,\ldots,K-1$, where the rows and columns associated with the vertices in $\mathcal{R}_S$ and $\mathcal{C}_S$ are ordered last as the coupling rows and linking columns, respectively. That is, a row vertex $r_i \in \mathcal{V}_k$ corresponds to permuting row i of A to the kth row slice $A^{\pi}_{k*} = (A^{\pi}_{k1} \ \cdots \ A^{\pi}_{kK} \ A^{\pi}_{kS})$ of A^{π}, and $r_i \in \mathcal{V}_S$ corresponds to permuting row i of A to the row border $A^{\pi}_{S*} = (A^{\pi}_{S1} \ \cdots \ A^{\pi}_{SK} \ A^{\pi}_{SS})$ of A^{π}. Similarly, a column vertex $c_j \in \mathcal{V}_k$ corresponds to permuting column j of A to the kth column slice $A^{\pi}_{*k} = \big((A^{\pi}_{1k})^T \ \cdots \ (A^{\pi}_{Kk})^T \ (A^{\pi}_{Sk})^T\big)^T$ of A^{π}, and $c_j \in \mathcal{V}_S$ corresponds to permuting column j of A to the column border $A^{\pi}_{*S} = \big((A^{\pi}_{1S})^T \ \cdots \ (A^{\pi}_{KS})^T \ (A^{\pi}_{SS})^T\big)^T$ of A^{π}.

Consider a row vertex $r_i \in \mathcal{R}_k$ and a column vertex $c_j \in \mathcal{C}_k$ of part $\mathcal{V}_k$ of a partition Π_{GPVS} of $\mathcal{B}_A$. Since $Adj(r_i) \subseteq \mathcal{C}_k \cup \mathcal{C}_S$, $r_i \in \mathcal{R}_k$ corresponds to permuting all nonzeros of row i of A into either submatrix A^{π}_{kk} or submatrices A^{π}_{kk} and A^{π}_{kS} depending on r_i being a non-boundary or a boundary vertex of $\mathcal{V}_k$, respectively. So, all nonzeros in the kth row slice A^{π}_{k*} of A^{π} will be confined to the A^{π}_{kk} and A^{π}_{kS} matrices. Since $Adj(c_j) \subseteq \mathcal{R}_k \cup \mathcal{R}_S$, $c_j \in \mathcal{C}_k$ corresponds to permuting all nonzeros of column j of A into either submatrix A^{π}_{kk} or submatrices A^{π}_{kk} and A^{π}_{Sk} of A^{π} depending on c_j being a non-boundary or a boundary vertex of $\mathcal{V}_k$, respectively. So, all nonzeros in the kth column slice A^{π}_{*k} of A^{π} will be confined to the A^{π}_{kk} and A^{π}_{Sk} matrices. Hence, A^{π} will be in a DB form as shown in (1.2) with $A^{\pi}_{kk} = B_k$, $A^{\pi}_{kS} = C_k$ and $A^{\pi}_{Sk} = R_k$, for $k = 1, \ldots, K$, and $A^{\pi}_{SS} = D$. The number of coupling rows and linking columns in A^{π} is equal to, respectively, the number of row and column vertices in the separator $\mathcal{V}_S$, i.e., $M_c = |\mathcal{R}_S|$ and $N_\ell = |\mathcal{C}_S|$. So, in GPVS of $\mathcal{B}_A$, minimizing the separator size according to (3.2) corresponds to minimizing the sum of the number of coupling rows and linking columns in A^{π}, since $|\mathcal{V}_S| = |\mathcal{R}_S| + |\mathcal{C}_S| = M_c + N_\ell$. The row and column dimensions of the kth diagonal block B_k of A^{π} is equal to, respectively, the number of row and column vertices in part $\mathcal{V}_k$, i.e., $M_k = |\mathcal{R}_k|$ and $N_k = |\mathcal{C}_k|$ for $k = 1, \ldots, K$. So, the row-vertex and column-vertex counts of the parts $\{\mathcal{V}_1, \ldots, \mathcal{V}_K\}$ can be used to maintain the required balance criterion on the dimensions of the diagonal blocks $\{B_1, \ldots, B_K\}$ of A^{π}. Fig. 4.2a displays a 3-way GPVS of $\mathcal{B}_A$, and Fig. 4.2b shows a corresponding partial permutation that transforms matrix A of Fig. 4.1 into a 3-way DB form A_{DB}.

5. Greedy Heuristics for Indirect GPVS.

In this section, we first propose a heuristic model for finding a better wide separator through GPES. Then, we propose a heuristic method to find a good vertex cover of a K-partite graph for wide-to-narrow separator refinement from K-way GPES. Finally, we present our critique on the optimality of the vertex-cover model on wide-to-narrow separator refinement.

5.1. Finding a Good Wide Vertex Separator.

Finding a good wide separator is an important and difficult step in indirect GPVS. In fact, the real difficulty is in the definition of the *goodness* for the wide-separator. There are no certain metrics for the goodness of a wide separator that will lead to a smaller narrow separator. Minimizing the number of edges on the cut may be a desirable metric, because it corresponds to finding better logical clusters of the given graph. Leiserson and Lewis [46] proposed minimizing the size of the wide separator as an alternative metric. Although both metrics are valuable assets for the goodness of a wide separator, they do not guarantee a narrow separator of smaller cardinality. For example, consider three different wide separators displayed in Fig. 5.1. The first one (Fig. 5.1a) has minimum number of edges, and the second one (Fig. 5.1b) has minimum number of vertices on the wide separator leading to narrow separators of size 3 and 2, respectively. The third one (Fig. 5.1c) has neither minimum number of edges nor minimum number of vertices, but it leads to a narrow separator of size 1.

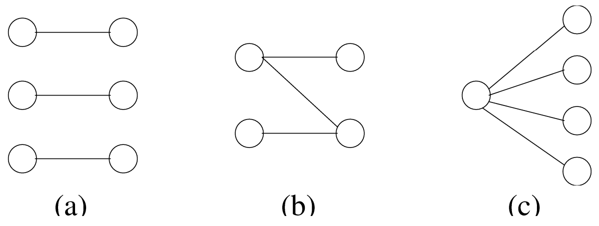


FIG. 5.1. *Three different wide vertex separators.*

Our basic observation is that all edges are not of equal importance for the goodness of a wide separator. Edges incident to a vertex with high degree are less important, since this vertex has a higher probability to be in a narrow separator. The example given in Fig. 5.1 supports this argument. For this purpose, we propose an edge weighting scheme. In this scheme, each edge of the graph is assigned a weight inversely proportional to the degrees of its end-vertices. That is, each edge $e_{ij}=(v_i, v_j)$ of the graph is assigned the weight

$$w_{ij} = \frac{1}{\max\{d_i, d_j\}} \tag{5.1}$$

where d_i and d_j denote the degrees of vertices v_i and v_j, respectively. Minimizing the cutsize of this edge-weighted graph according to (3.1) is expected to produce good wide separators for refining to narrow separators.

The general edge-weighting scheme in (5.1) can be used for A-to-A_{DB} transformation through indirect GPVS on bipartite graph representation $\mathcal{B}_A$ of matrix A. However, this edge-weighting scheme needs special attention if indirect GPVS is to be used in the first phase of a two-phase A-to-A_{SB} transformation. As mentioned in Section 3.3, a column vertex in the narrow separator of a GPVS of $\mathcal{B}_A$ may induce multiple coupling rows during the A_{DB}-to-A_{SB} transformation through the column-splitting method. So, in two-phase approaches, favoring row vertices to column vertices for the narrow separator can be expected to produce a better A_{SB}. This argument can be extended to favoring row vertices to column vertices in the wide separator to be obtained after GPES of $\mathcal{B}_A$. In this respect, we do not favor edges incident to a column vertex with a high degree to be on the cut of the GPES of $\mathcal{B}_A$. So, we propose considering only the degrees of the row vertices in the edge weighting of $\mathcal{B}_A$ for the two-phase approaches. That is, each edge $e_{ij}=(r_i \in \mathcal{R}, c_j \in \mathcal{C})$ of $\mathcal{B}_A=(\mathcal{V}=\mathcal{R}\cup\mathcal{C}, \mathcal{E})$ is assigned the weight

$$w_{ij} = \frac{1}{d_i} \tag{5.2}$$

where d_i denotes the degree of row vertex r_i.

5.2. Wide-to-Narrow Separator Refinement. A minimum vertex cover of a bipartite graph can be computed optimally in polynomial time by finding a maximum matching, by exploiting the duality between the two [45, 57, 58]. So, the wide-to-narrow separator refinement problem can be easily solved using this scheme for 2-way indirect GPVS, because the edge separator of a 2-way GPES induces a bipartite subgraph. This scheme has been widely exploited in a recursive manner in the nested-dissection based K-way indirect GPVS for ordering symmetric sparse matrices, because a 2-way GPES is adopted at each dissection step. In this work, we exploit this scheme to obtain a wide vertex separator directly from a K-way GPES in the A-to-A_{DB} transformation stage of our two-phase approach. Because, the bipartite graph model $\mathcal{B}_A$ for representing A in two-phase approaches ensures that the subgraph induced by any edge separator of $\mathcal{B}_A$ will also be a bipartite graph.

The minimum vertex cover problem is known to be NP-hard on K-partite graphs at least for $K \geq 5$ [16], thus we need to resort to heuristics. Leiserson and Lewis [46] proposed two greedy heuristics for this purpose, namely minimum recovery (MR) and maximum inclusion (MI). The MR heuristic is based on iteratively removing the vertex with minimum degree from the K-partite graph $\mathcal{K}(\mathcal{E}_S)$ and including all vertices adjacent to that vertex to the vertex cover $\mathcal{V}_S$. The MI heuristic is based on iteratively including the vertex with maximum degree into $\mathcal{V}_S$. In both heuristics, all edges incident to the vertices included into $\mathcal{V}_S$ are deleted from $\mathcal{K}(\mathcal{E}_S)$ so that the degrees of the remaining vertices in $\mathcal{K}(\mathcal{E}_S)$ are updated accordingly. Both heuristics continue the iterations until all edges are deleted from $\mathcal{K}(\mathcal{E}_S)$. In this work, we combine the advantages of these two heuristics in a new heuristic called one-recovery and maximum-inclusion (OR-MI). Each iteration of the OR-MI heuristic consists of two stages. In the first stage, all vertices of degree one are removed from $\mathcal{K}(\mathcal{E}_S)$ according to the MR heuristic, since this decision does not spoil our chance to find a minimum vertex cover. In the second stage, the vertex with maximum degree is included into $\mathcal{V}_S$ according to the MI heuristic. The details of these heuristics can be found in our previous work [56].

5.3. Vertex-Cover Model: On the Optimality of Separator Refinement. For $(\mathcal{K}=2)$-way GPES based GPVS, it was stated [58] that the minimum vertex cover $\mathcal{V}_S$ of the bipartite graph $\mathcal{K}(\mathcal{E}_S)=(\mathcal{V}_B=\mathcal{V}_{B1}\cup\mathcal{V}_{B2},\mathcal{E}_S)$ induced by an edge separator $\mathcal{E}_S$ of GPES $\Pi_{GPES} = (\mathcal{V}_1, \mathcal{V}_2)$ of $\mathcal{G}$ is a smallest vertex separator of $\mathcal{G}$ corresponding to $\mathcal{E}_S$. Recall that $\mathcal{V}_{Bk}$ denotes the set of boundary vertices of part $\mathcal{V}_k$. Here, we would like to discuss that this correspondence does not guarantee the optimality of the wide-to-narrow separator refinement. That is, the minimum vertex cover $\mathcal{V}_S$ of $\mathcal{K}(\mathcal{E}_S)$ may not constitute a minimum vertex separator that can be obtained from the wide separator $\mathcal{V}_B$. We can classify the boundary vertices $\mathcal{V}_{Bk}$ of a part $\mathcal{V}_k$ as *loosely-bound* and *tightly-bound* vertices. A loosely-bound vertex v_i of $\mathcal{V}_{Bk}$ is not adjacent to any non-boundary vertex of $\mathcal{V}_k$, i.e., $Adj(v_i,\mathcal{V}_k)=Adj(v_i)\cap\mathcal{V}_k\subseteq\mathcal{V}_{Bk}-\{v_i\}$, whereas a tightly-bound vertex v_j of V_{Bk} is adjacent to at least one non-boundary vertex of $\mathcal{V}_k$, i.e., $Adj(v_j,\mathcal{V}_k-\mathcal{V}_{Bk})\neq\emptyset$. Each cut edge between two tightly-bound vertices should always be covered by a vertex cover $\mathcal{V}_S$ of $\mathcal{K}(\mathcal{E}_S)$ for $\mathcal{V}_S$ to constitute a separator of $\mathcal{G}$. However, it is an unnecessarily severe measure to impose the same requirement for a cut edge incident to at least one loosely-bound vertex. If all vertices in $\mathcal{V}_{Bk}$ that are adjacent to a loosely-bound vertex $v_i \in \mathcal{V}_{Bk}$ are included into $\mathcal{V}_S$ then cut edges incident to v_i need not to be covered. For example, Fig. 5.2 illustrates a 2-way GPES, where $v_2 \in \mathcal{V}_{B1}$ is a loosely-bound vertex and all other vertices are tightly-bound vertices. Fig. 5.3(a) and Fig. 5.3(b) illustrate two optimal vertex covers $\mathcal{V}_S=\{v_1,v_2,v_3\}$ and $\mathcal{V}_S=\{v_1,v_6,v_7\}$, each of size 3, on bipartite graph $\mathcal{K}(\mathcal{E}_S)$. However, there is a wide-to-narrow separator refinement $\mathcal{V}_S=\{v_1,v_3\}$ of size 2 as shown in Fig. 5.3(c).

As mentioned in Section 8, Liu's narrow separator refinement algorithm [49] can also be considered as exploiting the vertex cover model on the bipartite graph induced by the edges between $\mathcal{V}_1$ and $\mathcal{V}_S$ ($\mathcal{V}_2$ and $\mathcal{V}_S$) of a GPVS $\Pi_{GPVS}=\{\mathcal{V}_1,\mathcal{V}_2;\mathcal{V}_S\}$. So, the discussion given here also applies to Liu's narrow separator refinement algorithm, where loosely-bound vertices can only exist in the $\mathcal{V}_1$ ($\mathcal{V}_2$) part of the bipartite graph.

The non-optimality of the minimum vertex–cover model has been overlooked probably due to the unlikeliness of loosely-bound vertices in the GPVS of graphs arising in finite difference and finite element applications.

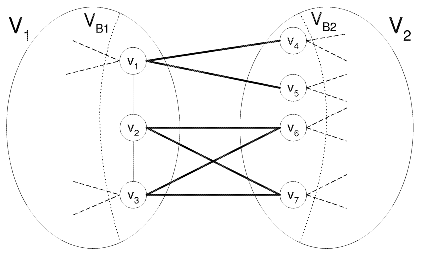


FIG. 5.2. *A sample 2-way GPES for wide-to-narrow separator refinement.*

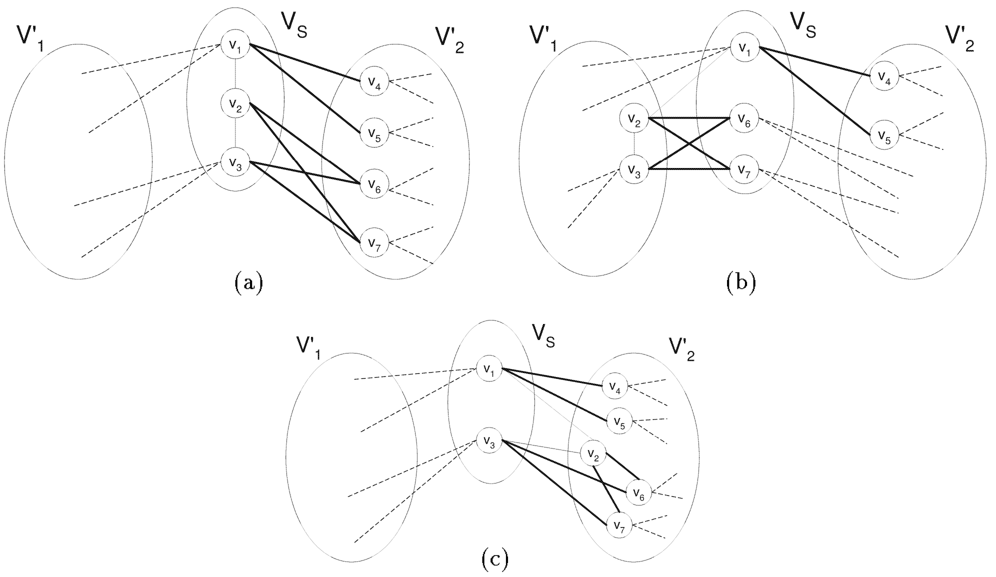
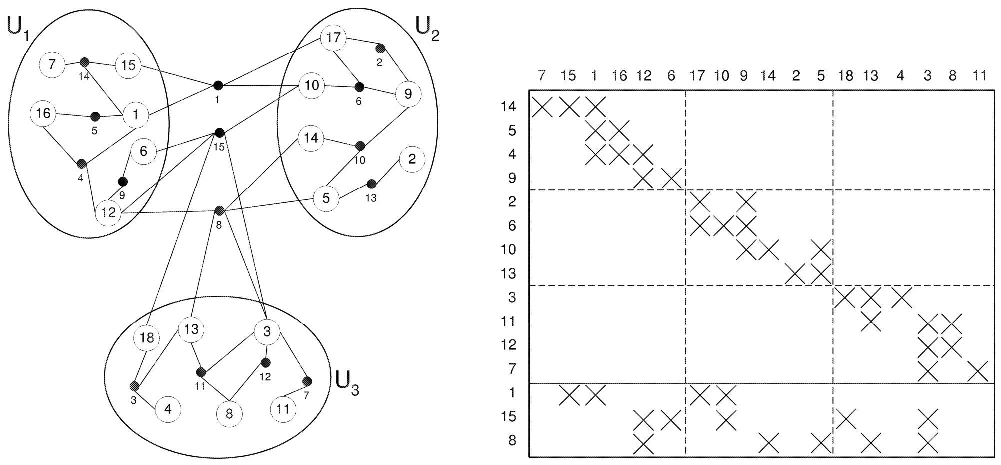


FIG. 5.3. *(a) & (b) Two wide-to-narrow separator refinements induced by two optimal vertex covers, (c) Optimal wide-to-narrow separator refinement.*

6. Hypergraph Model for A-to-A_{SB} Transformation.

In this section, we show that A-to-A_{SB} transformation can be described as an HP problem on a hypergraph representation of A. In our previous works [8, 9, 55, 56], we proposed two hypergraph models, namely row-net and column-net models, for representing rectangular as well as symmetric and nonsymmetric square matrices. These two models are duals of each other. The row-net representation of a matrix is equal to the column-net representation of its transpose and vice-versa. Here, we will only describe and discuss the row-net model for permuting a matrix A into a primal SB form, whereas the column-net model can be used for permuting A into a dual SB form. Because of the duality between the row-net and column-net models, permuting A into a dual SB form using the column-net model on A is the same as permuting A^T into a primal SB from using the row-net model on A^T.

In the (row-net) hypergraph model, an $M\times N$ matrix $A=(a_{ij})$ is represented as a hypergraph $\mathcal{H}_A=(\mathcal{U},\mathcal{N})$ on N nodes and M nets with the number of pins equal to the number of nonzeros in matrix A. Node and net sets $\mathcal{U}$ and $\mathcal{N}$ correspond, respectively, to the columns and rows of A. There exist one net n_i and one node u_j for each row i and column j, respectively. Net $n_i \subseteq \mathcal{U}$ contains the nodes corresponding to the columns that have a nonzero entry in row i, i.e., $u_j \in n_i$ if and only if $a_{ij} \neq 0$. That is, $Pins(n_i)$ effectively represents the set of columns that have a nonzero in row i of



(a) (b)

FIG. 6.1. *(a) Row-net hypergraph representation $\mathcal{H}_A$ of the sample A matrix shown in Fig. 4.1 and 3-way partitioning Π_{HP} of $\mathcal{H}_A$, (b) 3-way SB form A_{SB} of A induced by Π_{HP}.*

A, and in a dual manner $Nets(u_j)$ represents the set of rows that have a nonzero in column j of A. So, the size s_i of a net n_i is equal to the number of nonzeros in row i of A and the degree d_j of a node u_j is equal to the number of nonzeros in column j of A. Fig. 6.1(a) displays the hypergraph representation of the 16×18 sample matrix given in Fig. 4.1.

In our recent works [8, 9], we exploited the proposed row-net (column-net) model for columnwise (rowwise) decomposition of sparse matrices for parallel matrix-vector multiplication. In that application, nodes represent units of computation and nets encode multiway data dependencies. In [8, 9], we showed that 1D matrix partitioning problem can be modeled as an HP problem in which the cutsize metric given in (3.3.b) is exactly equal to the parallel communication volume. The proposed HP model overcomes the major flaws and limitations of the standard GPES models, which are also addressed by Hendrickson and Kolda [27, 30]. In this work, we show that the A-to-A_{SB} transformation problem can be described as an HP problem in which the cutsize metric given in (3.3.a) is exactly equal to the number of coupling rows in A_{SB}.

Consider a K-way partition $\Pi_{HP} = \{\mathcal{U}_1, \ldots, \mathcal{U}_K\} = \{\mathcal{N}_1, \ldots, \mathcal{N}_K; \mathcal{N}_S\}$ of $\mathcal{H}_A$. Π_{HP} can be decoded as a partial permutation on the rows and columns of A to induce a permuted matrix A^π. In this permutation, the columns associated with the nodes in $\mathcal{U}_{k+1}$ are ordered after the columns associated with the nodes in $\mathcal{U}_k$, for $k = 1, \ldots, K-1$. The rows associated with the internal nets ($\mathcal{N}_{k+1}$) of $\mathcal{U}_{k+1}$ are ordered after the rows associated with the internal nets ($\mathcal{N}_k$) of $\mathcal{U}_k$, for $k = 1, \ldots, K-1$, where the rows associated with the external nets ($\mathcal{N}_S$) are ordered last as the coupling rows. That is, a node $u_j \in \mathcal{U}_k$ corresponds to permuting column j of A to the kth column slice $A^\pi_{*k} = \big((A^\pi_{1k})^T \cdots (A^\pi_{Kk})^T\ (A^\pi_{Sk})^T\big)^T$ of A^π. An internal net n_i of $\mathcal{U}_k$ corresponds to permuting row i of A to the kth row slice $A^\pi_{k*} = (A^\pi_{k1} \cdots A^\pi_{kK})$ of A^π, and an external net n_i corresponds to permuting row i of A to the border $A^\pi_S = (A^\pi_{S1} \cdots A^\pi_{SK})$ of A^π.

Consider an internal net $n_i \in \mathcal{N}_k$ of part $\mathcal{U}_k$ of a partition Π_{GPVS} of $\mathcal{H}_A$. Since $Pins(n_i) \subseteq \mathcal{U}_k$, $n_i \in \mathcal{N}_k$ corresponds to permuting all nonzeros of row i of A into submatrix A^π_{kk} of A^π. So, all nonzeros in the kth row slice A^π_{k*} will be confined to the A^π_{kk} submatrix. Consider a node u_j of part $\mathcal{U}_k$. Since $Nets(u_j) \subseteq \mathcal{N}_k \cup \mathcal{N}_S$, $u_j \in \mathcal{U}_k$ corresponds to permuting all nonzeros of column j of A into either submatrix A^π_{kk} or submatrices A^π_{kk} and A^π_{kS} depending on u_j being a non-boundary or a boundary node of $\mathcal{U}_k$, respectively. So, all nonzeros in the kth column slice A^{π}_{*k} will be confined to the A^{π}_{kk} and A^{π}_{Sk} matrices. Hence, A^{π} will be in an SB form as shown in (1.1) with $A^{\pi}_{kk}=B_k$ and $A^{\pi}_{Sk}=R_k$, for $k=1,\ldots,K$. The number of coupling rows in A^{π} is equal to the number of external nets, thus minimizing the cutsize according to (3.3.a) corresponds to minimizing the number of coupling rows in A^{π}. The row and column dimensions of the kth diagonal block B_k of A^{π} is equal to, respectively, the number of internal nets and nodes in part $\mathcal{U}_k$, i.e., $M_k=|\mathcal{N}_k|$ and $N_k=|\mathcal{U}_k|$ for $k=1,\ldots,K$. So, the node and internal-net counts of the parts $\{\mathcal{U}_1,\ldots,\mathcal{U}_K\}$ can be used to maintain the required balance criterion on the dimensions of the diagonal blocks $\{B_1,\ldots,B_K\}$ of A^{π}. Fig. 6.1(a) displays a 3-way partitioning Π_{HP} of $\mathcal{H}_A$ and Fig. 6.1(b) shows a corresponding partial permutation which transforms A given in Fig. 4.1 directly into a 3-way SB form.

7. Formulating HP Problem as a GPVS Problem. In this section, we first summarize and criticize the previous work on alternative models proposed in the VLSI community for solving the HP problem. Then, we describe our novel and accurate GPVS-based formulation for the HP problem. Finally, we provide a discussion on the matrix theoretical view of the relation between HP and GPVS problems.

7.1. Alternative Models for Solving HP Problem. As indicated in the survey by Alpert and Kahng [1], hypergraphs are commonly used to represent circuit netlist connections in solving the circuit partitioning and placement problems in VLSI layout design. The circuit partitioning problem is to divide a system specification into clusters to minimize inter-cluster connections. Other circuit representation models were also proposed and used in the VLSI literature including dual hypergraph, clique-net graph (CNG) and net-intersection graph (NIG) [1]. Hypergraphs represent circuits in a natural way so that the circuit partitioning problem is directly described as an HP problem. Thus, these alternative models can be considered as alternative approaches for solving the HP problem.

The dual of a hypergraph $\mathcal{H}=(\mathcal{U},\mathcal{N})$ is defined as a hypergraph $\mathcal{H}'$, where the nodes and nets of $\mathcal{H}$ become, respectively, the nets and nodes of $\mathcal{H}'$. That is, $\mathcal{H}'=(\mathcal{U}',\mathcal{N}')$ with $Nets(u'_i)=Pins(n_i)$ for each $u'_i\in\mathcal{U}'$ and $n_i\in\mathcal{N}$, and $Pins(n'_j)=Nets(u_j)$ for each $n'_j\in\mathcal{N}'$ and $u_j\in\mathcal{U}$.

In the CNG model, the vertex set of the target graph is equal to the node set of the given hypergraph. Each net of the given hypergraph is represented by a clique of vertices corresponding to its pins. The multiple edges connecting each pair of vertices of the graph are contracted into a single edge, the weight of which is equal to the sum of the weights of the edges it represents. If an edge is in the cut set of a GPES then all nets represented by this edge are in the cut set of hypergraph partitioning, and vice versa. Ideally, no matter how nodes 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 CNG representation is that it is impossible to achieve such a *perfect* weighting of the edges as proved by Ihler et al. [34].

In the NIG representation $\mathcal{G}=(\mathcal{V},\mathcal{E})$ of a given hypergraph $\mathcal{H}=(\mathcal{U},\mathcal{N})$, each vertex v_i of $\mathcal{G}$ corresponds to net n_i of $\mathcal{H}$. Two vertices $v_i,v_j\in\mathcal{V}$ of $\mathcal{G}$ are adjacent if and only if respective nets $n_i,n_j\in\mathcal{N}$ of $\mathcal{H}$ share at least one pin, i.e., $e_{ij}\in\mathcal{E}$ if and only if $Pins(n_i)\cap Pins(n_j)\neq\emptyset$. So,

$$Adj(v_i)=\{v_j\ :\ n_j\in\mathcal{N}\ \ni\ Pins(n_i)\cap Pins(n_j)\neq\emptyset\}. \qquad (7.1)$$

Note that for a given hypergraph $\mathcal{H}$, NIG $\mathcal{G}$ is well-defined, however there is no unique reverse construction [1].

Both dual hypergraph and NIG models view the HP problem in terms of partitioning nets instead of nodes. Kahng [37] and Cong, Hagen, and Kahng [11] exploited this perspective of the NIG model to formulate the hypergraph bipartitioning problem as a two-stage process. In the first stage, nets of $\mathcal{H}$ are bipartitioned through 2-way GPES of its NIG $\mathcal{G}$. The resulting net bipartition induces a partial node bipartition on $\mathcal{H}$. Because, the nodes (pins) that belong only to the nets on one side of the bipartition can be unambigiuosly assigned to that side. However, other nodes may belong to the nets on both sides of the bipartition. Thus, the second stage involves finding the best completion of the partial node bipartition; i.e., a part assignment for the shared nodes such that the cutsize (3.3) is minimized. This problem is known as the module (node) contention problem in the VLSI community. Kahng [37] used a winner-loser heuristic [23], whereas Cong et al. [11] used a matching-based (IG-match) algorithm for solving the 2-way module contention problem optimally. Cong, Labio, and Shivakumar [12] extended this approach to K-way HP through using the dual hypergraph model. In the first stage, a K-way net partitioning is obtained through partitioning the dual hypergraph. For the second stage, they formulated the K-way module contention problem as a min-cost max-flow problem through defining binding factors between nodes and nets, and a preference function between parts and nodes.

Here, we reveal the fact that the module contention problem encountered in the second stage of the NIG-based hypergraph bipartitioning approaches [11, 37] is similar to the wide-to-narrow separator refinement problem encountered in the second stage of the indirect GPVS approaches widely used in nested-dissection based low-fill orderings for sparse matrix factorization. The module contention and separator refinement algorithms effectively work on the bipartite graph induced by the cut edges of a two-way GPES of the NIG representation of hypergraphs and the standard graph representation of sparse matrices, respectively. The winner-loser assignment heuristic [23, 37] used by Kahng [37] is very similar to the minimum-recovery heuristic proposed by Leiserson and Lewis [46] for separator refinement. Similarly, the IG-match algorithm proposed by Cong et al. [11] is similar to the maximum-matching based minimum vertex-cover algorithm [45, 57] used by Pothen, Simon, and Liou [58] for separator refinement. Despite not being stated in the literature, these net-bipartitioning based HP algorithms using the NIG model can be viewed as trying to solve the HP problem through an indirect GPVS of the NIG representation.

7.2. An Accurate Formulation of HP as GPVS on NIG Model. In this work, we propose a net-partitioning based K-way HP algorithm that avoids the module contention problem by showing that the HP problem can be described as a GPVS problem through the NIG model. A K-way vertex partition $\Pi_{GPVS} = \{\mathcal{V}_1, \ldots, \mathcal{V}_K; \mathcal{V}_S\}$ of NIG $\mathcal{G}$ by a vertex separator $\mathcal{V}_S$ can be decoded as inducing a $(K+1)$-way net partition $\{\mathcal{N}_1, \ldots, \mathcal{N}_K; \mathcal{N}_S\}$ on $\mathcal{H}$, where $\mathcal{N}_k = \mathcal{V}_k$ for $k = 1, \ldots, K$ and $\mathcal{N}_S = \mathcal{V}_S$. By definition of GPVS, we have $Adj(\mathcal{V}_k) \cap \mathcal{V}_\ell = \emptyset$ for $1 \leq k < \ell \leq K$. This implies that $Pins(\mathcal{N}_k) \cap Pins(\mathcal{N}_\ell) = \emptyset$ for $1 \leq k < \ell \leq K$. Because, if any two nets $n_i \in \mathcal{N}_k$ and $n_j \in \mathcal{N}_\ell$ shared at least one pin, then there would be an edge e_{ij} between vertices $v_i \in \mathcal{V}_k$ and $v_j \in \mathcal{V}_\ell$ in $\mathcal{G}$, which would correspond to an edge between parts $\mathcal{V}_k$ and $\mathcal{V}_\ell$ of Π_{GPVS} contradicting the definition of GPVS. Thus, net partition $\{\mathcal{N}_1, \ldots, \mathcal{N}_K; \mathcal{N}_S\}$ induces a K-way contention-free, partial node partition

$$\Pi'_{HP} = \{\mathcal{U}'_1 = Pins(\mathcal{N}_1), \ldots, \mathcal{U}'_K = Pins(\mathcal{N}_K)\}. \tag{7.2}$$

Since $\mathcal{U}'_k = Pins(\mathcal{N}_k)$, $\mathcal{N}_k$ corresponds to the set of internal nets of part $\mathcal{U}'_k$ for $k = 1, \ldots, K$. So, $\mathcal{N}_S = \mathcal{V}_S$ constitutes a net cut for Π'_{HP}. The set $\mathcal{U}'_F$ of remaining nodes satisfies $Nets(\mathcal{U}'_F) \subseteq \mathcal{N}_S$ since

$$(7.3) \qquad \mathcal{U}'_F = \mathcal{U} - \bigcup_{k=1}^{K} \mathcal{U}'_k = \mathcal{U} - \bigcup_{k=1}^{K} Pins(\mathcal{N}_k) = \{u_i \in \mathcal{U} : Nets(u_i) \subseteq \mathcal{N}_S\}.$$

The nodes in $\mathcal{U}'_F$ are referred to here as the free nodes, because they can be freely assigned to any part of Π'_{HP} without disturbing net cut $\mathcal{N}_S$. Consider any arbitrary assignment of the nodes in $\mathcal{U}'_F$ to the parts of Π'_{HP} to obtain a complete K-way partition $\Pi_{HP} = \{\mathcal{U}_1, \ldots, \mathcal{U}_K\}$. Since $\mathcal{N}_S$ is a net cut of Π'_{HP}, the assignment of the nodes of $\mathcal{U}'_F$ cannot remove any net from the cut. Furthermore, since $Nets(\mathcal{U}'_F) \subseteq \mathcal{N}_S$, all nets of the nodes of $\mathcal{U}'_F$ are already in the net cut of Π'_{HP}. So, this arbitrary assignment will not introduce any additional nets to the cut of Π_{HP}. Note that the free nodes can easily be identified in Π_{HP} as the the boundary nodes which are not connected to any internal nets. Thus, Π_{HP} will be a K-way partition of $\mathcal{H}$ with cutsize $|\mathcal{N}_S|$ (3.3.a) equal to the separator size $|\mathcal{V}_S|$ (3.2) of Π_{GPVS} of NIG $\mathcal{G}$.

Fig. 7.1(a) displays the NIG representation $\mathcal{G}$ of a sample hypergraph $\mathcal{H}$ shown in Fig. 6.1(a). Fig. 7.1(a) also displays a 3-way GPVS Π_{GPVS} of $\mathcal{G}$. Fig. 6.1(a) shows the 3-way partitioning Π_{HP} of $\mathcal{H}$ induced by Π_{GPVS} of $\mathcal{G}$ with $|\mathcal{N}_S| = |\mathcal{V}_S| = 3$. Note that Π_{HP} shown in Fig. 6.1(a) contains no free nodes since each boundary node is connected to at least one internal net.

The above mentioned equivalence between the HP and GPVS problems is a one way equivalence in practice since there is no unique reverse hypergraph construction from a given graph [1]. Furthermore, the proposed formulation of the K-way HP problem as a K-way GPVS problem is not valid for the connectivity cutsize metric given in (3.3.b).

In the proposed formulation of the HP problem as a GPVS problem the NIG model suffers from information loss on the hypergraph nodes (H-nodes). Hence, a given balance criterion on the nodes of the hypergraph cannot be enforced during the GPVS of the respective NIG. A straightforward solution to avoid this limitation of the NIG model is to maintain the hypergraph together with the NIG during the GPVS operation. However, this straightforward approach will suffer from both storage and run-time inefficiency. Here, we propose an approximate model for estimating the number of H-nodes in each part of a partition of the NIG. As mentioned earlier, the NIG model can also be viewed as a clique-node model since each node of the hypergraph induces an edge between every net it is connected to. So, the edges of the NIG implicitly represent the nodes of the hypergraph, where each H-node u_h of degree d_h induces $\binom{d_h}{2}$ edges in the NIG. We assign a uniform weight of $1/\binom{d_h}{2}$ to every clique edge induced by H-node u_h. That is, we assign a weight w_{ij} to each edge e_{ij} of the NIG, where

$$(7.4) \qquad w_{ij} = \sum_{u_h \in (n_i \cap n_j)} \frac{1}{\binom{d_h}{2}}$$

Consider a GPVS $\Pi_{GPVS} = \{\mathcal{V}_1, \ldots, \mathcal{V}_K; \mathcal{V}_S\}$ of NIG $\mathcal{G}$ representing a given hypergraph $\mathcal{H}$. The sum W_k of the weights of the internal and external edges of each part $\mathcal{V}_k$ in Π_{GPVS} will approximate the H-node count of part $\mathcal{U}'_k$ in the corresponding partial partition $\Pi'_{HP} = \{\mathcal{U}'_1, \ldots, \mathcal{U}'_K\}$ of $\mathcal{H}$. Edge-weight sum W_k of part $\mathcal{V}_k$ in Π_{GPVS} will model the correct count for the non-boundary H-nodes of part $\mathcal{U}'_k$, and even for the boundary H-nodes of $\mathcal{U}'_k$ that are connected to only one cut net. Only the boundary H-nodes that are connected to more than one cut net will introduce errors. Consider a boundary H-node u_h of part $\mathcal{U}'_k$ with an external degree $\delta_h < d_h$, i.e., u_h is connected to δ_h cut nets. Then, u_h will contribute by an amount of $1 - \binom{\delta_h}{2}/\binom{d_h}{2}$ to W_k instead of 1. So, edge-weight sum W_k of part $\mathcal{V}_k$ in Π_{GPVS} will be less than

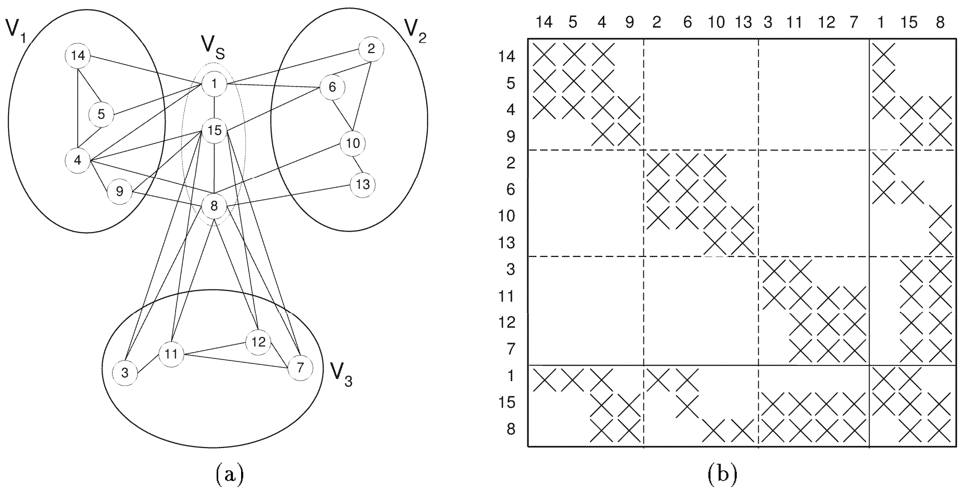


FIG. 7.1. *(a) Net-intersection graph representation $\mathcal{G}_{NIG}(\mathcal{H}_A)$ of hypergraph $\mathcal{H}_A$ shown in Fig. 6.1(a) and 3-way partitioning Π_{GPVS} of $\mathcal{G}_{NIG}(\mathcal{H}_A)$ by vertex separator, (b) 3-way DB form of AA^T induced by Π_{GPVS}.*

the actual H-node count of part $\mathcal{U}'_k$ in Π'_{HP}. As the edge-weight sums of different parts will involve similar errors, the proposed scheme can be expected to produce a sufficiently good balance on the H-node counts of the parts of Π'_{HP}. The free nodes in $\mathcal{U}'_F$ can easily be exploited to improve the balance during the completion of partial partition Π'_{HP} to Π_{HP}.

7.3. Matrix Theoretical View of the Relation Between HP and GPVS. Let $\mathcal{G}_{NIG}(\mathcal{H}_A)=(\mathcal{V},\mathcal{E})$ denote the NIG model for the row-net hypergraph representation $\mathcal{H}_A=(\mathcal{U},\mathcal{N})$ of matrix A. By definition of the NIG model, the vertices of $\mathcal{G}_{NIG}$ will represent the rows of A, and $e_{ij}\in\mathcal{E}$ if and only if $Pins(n_i)\cap Pins(n_j)\neq\emptyset$. Since $Pins(n_i)$ represents the set of columns that have a nonzero in row i of A, $Pins(n_i)\cap Pins(n_j)\neq\emptyset$ corresponds to the condition that rows i and j of A have a nonzero in at least one common column. Let $Z=(z_{ij})$ denote the $M\times M$ matrix $Z=AA^T$. Since $z_{ij}=<r_i,r_j^T>$, z_{ij} will be nonzero if and only if $e_{ij}\in\mathcal{E}$. Hence, the sparsity pattern of symmetric matrix Z will correspond to the adjacency matrix representation of $\mathcal{G}_{NIG}$. In other words, $\mathcal{G}_{NIG}$ will be equivalent to the standard graph representation of symmetric matrix Z, i.e., $\mathcal{G}_{NIG}(\mathcal{H}_A)\equiv\mathcal{G}_{AA^T}$. Note that although vertex v_i of $\mathcal{G}_{NIG}$ represents only row i of A, it represents both row i and column i of AA^T in $\mathcal{G}_{AA^T}$. A dual equivalence holds for the column-net hypergraph model of A so that the NIG representation of the column-net model of A is equivalent to the standard graph representation of symmetric matrix A^TA.

Fig. 7.1(a) displays the NIG representation $\mathcal{G}_{NIG}(\mathcal{H}_A)$ of the row-net hypergraph model $\mathcal{H}_A$ given in Fig. 6.1(a) for the sample A-matrix in Fig. 4.1. Note that $\mathcal{G}_{NIG}(\mathcal{H}_A)$ corresponds to the standard graph representation of AA^T shown in Fig. 7.1(b). The discussion given in Section 4 for bipartite graphs can easily be extended to show that the problem of transforming a symmetric matrix into a DB form through symmetric row/column permutation can be modeled as a GPVS problem on its standard graph representation. So, Fig. 7.1(b) shows a 3-way DB form of the AA^T matrix induced by the 3-way GPVS Π_{GPVS} of $\mathcal{G}_{NIG}(\mathcal{H}_A)$ shown in Fig. 7.1(a). Recall that 3-way partitioning Π_{HP} shown in Fig. 6.1(a) is induced by Π_{GPVS} of $\mathcal{G}_{NIG}(\mathcal{H}_A)$. Hence, Π_{GPVS} induces the same SB form A_{SB} of A as shown in Fig. 6.1(b).

8. Graph and Hypergraph Partitioning Algorithms and Tools. Recently, *multilevel* GPES [7, 21, 25, 41, 42] and HP [8, 9, 24, 39] approaches have been proposed leading to successful GPES tools Chaco [26], MeTiS [38] and WGPP [22], and HP tools hMeTiS [40] and PaToH [10]. 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/hypergraph by adopting various matching heuristics until the number of vertices in the coarsened graph/hypergraph reduces below a predetermined threshold value. Clustering corresponds to coalescing highly interacting vertices to supernodes. In the second phase, a partition is obtained on the coarsest graph/hypergraph using various heuristics including FM, which is an iterative refinement heuristic proposed for graph/hypergraph partitioning by Fiduccia and Mattheyses [15] as a faster implementation of the KL algorithm proposed by Kernighan and Lin [43]. In the third phase, the partition found in the second phase is successively projected back towards the original graph/hypergraph by refining the projected partitions on the intermediate level uncoarser graphs/hypergraphs using various heuristics including FM. In this work, we use direct K-way GPES version of MeTiS [41] (*kmetis* option [38]) in our indirect GPVS algorithms and our multilevel HP tool PaToH [10] in our one-phase A-toA_{SB} transformation approach.

One of the most important applications of GPVS is George's *nested–dissection* algorithm [18, 19], which has been widely used for reordering of the rows/columns of a symmetric, sparse, and positive definite matrix to reduce *fill* in the factor matrices. Here, GPVS is defined on the standard graph model of the given symmetric matrix. The basic idea in the nested dissection algorithm is to reorder symmetric matrix into a 2-way DB form so that no fill can occur in the off-diagonal blocks. The DB form of the given matrix is obtained through a symmetric row/column permutation induced by a 2-way GPVS. Then, both diagonal blocks are reordered by applying the dissection strategy recursively. The performance of the nested-dissection reordering algorithm depends on finding small vertex separators at each dissection step. So, the nested-dissection implementations can easily be exploited for obtaining a K-way DB form of a matrix by terminating the dissection operation after $\lg_2 K$ recursion levels and then gathering the vertex separators obtained at each dissection step to a single separator constituting a K-way vertex separator. So, we obtain K-way DB forms of A and AA^T matrices in our two-phase and one-phase approaches by providing the bipartite graph and net-intersection graph models, respectively, as inputs to a nested-dissection based reordering tool. Note that in the former approach we effectively perform a nonsymmetric nested dissection on the bipartite graph model of the rectangular A matrix.

The multilevel GPES approaches have been used in several multilevel nested–dissection implementations based on indirect 2-way GPVS. In this work, we use the *oemetis* ordering code of MeTiS [38] to compare against our edge-weighted indirect GPVS algorithms. Recently, direct 2-way GPVS approaches have been embedded into various multilevel nested-dissection implementations [22, 28, 38]. In these implementations, a 2-way GPVS obtained on the coarsest graph is refined during the multilevel framework of the uncoarsening phase. Two distinct vertex-separator refinement schemes were proposed and used for the uncoarsening phase. The first one is the extension of the FM edge-separator refinement approach to vertex-separator refinement as proposed by Ashcraft and Liu [2]. This scheme considers vertex moves from vertex separator $\mathcal{V}_S$ to both $\mathcal{V}_1$ and $\mathcal{V}_2$ in $\Pi_{GPVS} = \{\mathcal{V}_1, \mathcal{V}_2; \mathcal{V}_S\}$. This refinement scheme is adopted in the *onmetis* ordering code of MeTiS [38], ordering code of

WGPP [22], and the ordering code BEND [28]. The second scheme is based on Liu's narrow separator refinement algorithm [49], which considers moving a set of vertices simultaneously from $\mathcal{V}_S$ at a time, in contrast to the FM-based refinement scheme [2], which moves only one vertex at a time. Liu's refinement algorithm [49] can be considered as repeatedly running the maximum-matching based vertex cover algorithm on the bipartite graphs induced by the edges between $\mathcal{V}_1$ and $\mathcal{V}_S$, and $\mathcal{V}_2$ and $\mathcal{V}_S$. That is, the wide vertex separator consisting of $\mathcal{V}_S$ and the boundary vertices of $\mathcal{V}_1$ ($\mathcal{V}_2$) is refined as in the GPES-based wide-to-narrow separator refinement scheme. The network-flow based minimum weighted vertex cover algorithms proposed by Ashcraft and Liu [3], and Hendrickson and Rothberg [28] enabled the use of Liu's refinement approach [49] on the coarse graphs within the multilevel framework. In this work, we use the publicly available *onmetis* ordering code of MeTiS [38] for direct GPVS.

9. Experimental Results. We have tested the performance of the proposed models and associated solution approaches on a wide range of large LP constraint matrices obtained from [33] and [36]. Properties of these rectangular matrices are presented in Table 9.1, where the matrices are listed in the order of increasing number of rows. For each $M \times N$ rectangular A matrix, properties of the AA^T matrix are displayed as well, since the NIG model of the row-net hypergraph representation of A corresponds to the standard graph representation of AA^T.

TABLE 9.1
Properties of rectangular test matrices.

	Rectangular matrix A								AA^T		
	number of		number of nonzeros						number nonzeros		
name	rows	cols	total	per row		per col			total	per row/col	
	M	N		max	avg	max	avg			max	avg
NL	7039	9718	41428	149	5.89	15	4.26		105089	360	13.93
CQ9	9278	13778	88897	390	9.58	24	6.45		221590	701	22.88
GE	10099	11098	39554	47	3.92	36	3.56		112129	114	10.10
CO9	10789	14851	101578	440	9.41	28	6.84		249205	706	22.10
car4	16384	33052	63724	111	3.89	109	1.93		199008	400	11.15
fxm4-6	22400	30732	248989	57	11.12	24	8.10		526536	97	22.51
fome12	24284	48920	142528	228	5.87	14	2.91		329068	567	12.55
pltexpA4-6	26894	70364	143059	30	5.32	8	2.03		269736	203	9.03
kent	31300	16620	184710	960	5.90	18	11.11		839040	2325	25.81
world	34506	32734	164470	341	4.77	16	5.02		582064	971	15.87
mod2	34774	31728	165129	310	4.75	16	5.20		604910	940	16.40
lpl1	39951	125000	381259	177	9.54	16	3.05		541217	465	12.55
fxm3-16	41340	64162	370839	57	8.97	36	5.78		765526	157	17.52

All experiments were performed on a workstation equipped with a 133 MHz PowerPC processor with 512-KB external cache and 64-MB of memory. We have tested $K = 4$, 8, and 16 way partitioning of every test matrix. For each K value, K-way partitioning of a test matrix constitutes a partitioning instance. Partitioning tools MeTiS [38] and PaToH [10] were run 50 times for each instance. We use averages of these runs for each instance in this section. Fig. 9.1 displays $K = 4$, 8, and 16 way sample primal SB forms of the GE matrix obtained by PaToH.

As would be expected the breadth of this work led to an enormous amount of experimental work. Here we will present only a portion of our experiments for the clarity of the main results of this work. The organization of this section is as follows. We first compare different solution techniques for a model. Tables 9.2–9.4 present only the averages over the 13 matrices. Breakdown of the results for each matrix can be

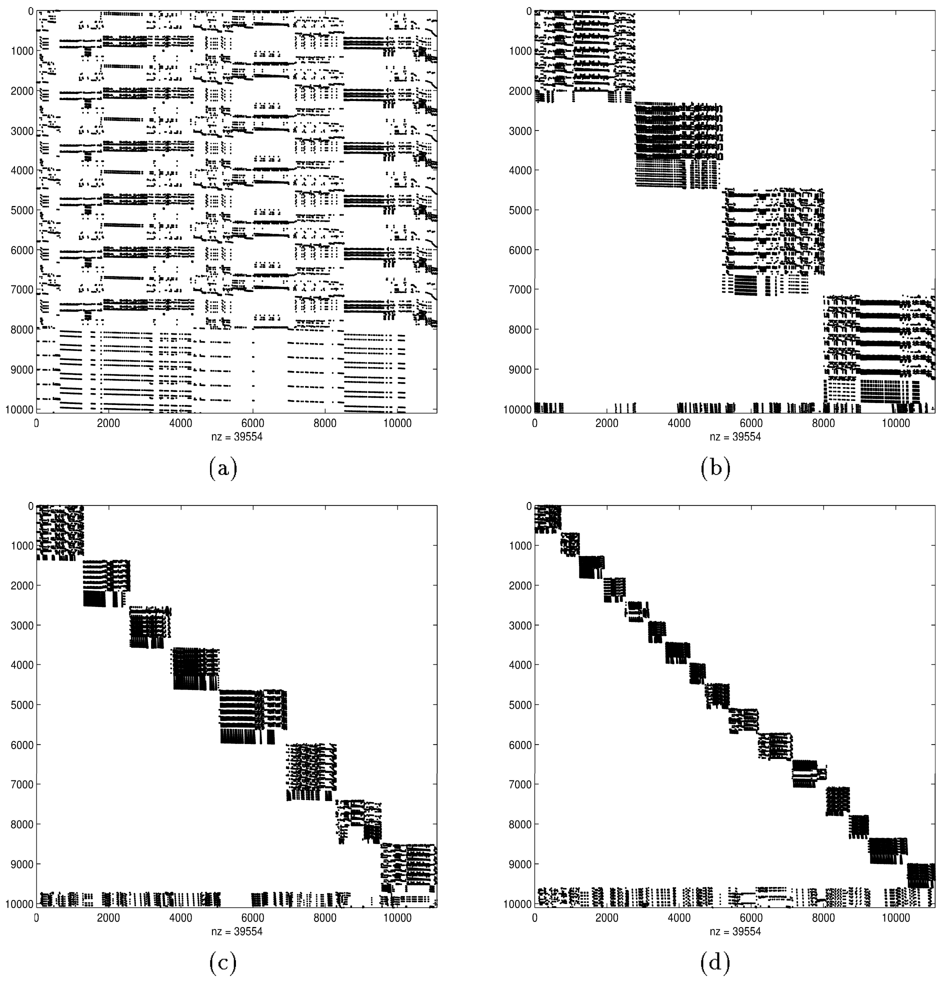


FIG. 9.1. *Rectangular GE matrix with 10098 rows and 11098 columns: (a) original structure, (b) 4-way SB form, (c) 8-way SB form, (d) 16-way SB form.*

found in [5]. Finally, we compare the effectiveness of the models for their best solution technique, both in terms of solution quality (Tables 9.5–9.6) and pre-processing times (Table 9.6).

In the tables, $\%M_c$ denotes the percentage of the number of coupling rows in both DB and primal SB forms, i.e., $\%M_c = 100{\times}M_c/M$. $\%N_\ell$ denotes the number of linking columns in the DB forms as percents of the respective M values to enable the comparison of the M_c and N_ℓ values under the same unit, i.e., $\%N_\ell = 100{\times}N_\ell/M$. We measure the balance quality of the diagonal blocks in terms of percent row imbalance $\%RI = 100{\times}(M_{max}/M_{avg} - 1)$ and percent column imbalance $\%CI = 100{\times}(N_{max}/N_{avg} - 1)$. Here, M_{max} (N_{max}) denotes the row (column) count of the diagonal block with the maximum number of rows (columns) in both SB and DB forms. $M_{avg} = (M - M_c)/K$ in both SB and DB forms, whereas $N_{avg} = (N - N_\ell)/K$ in DB forms and $N_{avg} = N/K$ in SB forms. It should be noted here that more complicated balancing criteria might need to be maintained in practical applications. For example, empirical relation $T(M,N) = cM^{2.17}N^{0.89}$ was reported by Medhi [50] for

Table 9.2
Performance of different techniques on the bipartite-graph model.

K	Indirect GPVS						Direct GPVS		
	BG-model (FH)			wBG-model			BG-model (*onmetis*)		
	A_{DB}		A_{SB}	A_{DB}		A_{SB}	A_{DB}		A_{SB}
	%M$_c$	%N$_\ell$	%M$_c$	%M$_c$	%N$_\ell$ (%N$'_\ell$)	%M$_c$	%M$_c$	%N$_\ell$	%M$_c$
4	6.55	0.20	6.80	1.51	1.02 (1.07)	2.56	1.31	0.22	1.60
8	9.70	0.54	10.40	2.56	2.01 (2.11)	4.65	2.75	0.65	3.60
16	12.79	1.05	14.12	3.82	3.13 (3.28)	7.29	4.15	1.17	5.90
Avg	9.68	0.60	10.44	2.63	2.05 (2.15)	4.83	2.74	0.68	3.70

the solution time (with IMSL routine ZX0LP [35]) of an LP subproblem corresponding to an $M \times N$ block diaogonal, where c is some constant. Although our HP tool PaToH can be modified to handle such balancing criteria, it is hard to impose a balancing criterion, which involves the product of M and N, into the existing GP tools.

Table 9.2 presents the results of our experiments on the bipartite graph model for both A-to-A_{DB} transformation and two-phase A-to-A_{SB} transformation. On the BG model, we experimented with the built-in GPES tool *kmetis* of MeTiS for indirect GPVS approaches and direct GPVS tool *onmetis*. Note that FH corresponds to our implementation of the algorithm proposed by Ferris and Horn [14], where we used *kmetis* to partition the bipartite graph. In our weighted-BG (wBG) scheme, the edges of the bipartite-graph are weighted according to (5.2) and this edge-weighted BG is provided to *kmetis*. Then the resulting GPES partition is converted to a GPVS partition by using the maximum-matching based exact vertex-cover algorithm for solving the K-way wide-to-narrow separator refinement problem through exploiting the bipartite nature of the graph. Since the GPES and GPVS solvers of MeTiS maintain balance on vertices, balance on the sum of the row and column counts of the diagonal blocks is explicitly maintained during partitioning. All three schemes produce DB forms with comparable row and column imblance values. As seen in Table 9.2, the proposed indirect wBG scheme (5.2) produces substantially better DB forms than the indirect FH scheme while approaching to the quality of direct *onmetis* scheme. It is interesting to note that the BG-onmetis scheme produces DB forms with wide row borders and narrow column borders in general, whereas the wBG scheme produces DB forms with comparable row and column border sizes in general.

For the results of our wBG scheme, an additional $\%N'_\ell$ column in parenthesis is displayed in Table 9.2 to experimentally verify our observation in Section 5.3. Here, N'_ℓ denotes the number of linking columns obtained after running the minimum vertex-cover algorithm, and N_ℓ denotes the number of linking columns obtained after removing the linking columns of connectivity 1 from the column borders of the DB forms. This post-processing simply corresponds to identifying and removing vertex v_2 from $\mathcal{V}_S$ in Fig. 5.3(a). So, $(N'_\ell - N_\ell)/(M_c + N'_\ell)$ effectively denotes the ratio of the number of redundant column vertices in the separator to the total number of separator vertices. On average, 2.1% of the separator vertices are identified to be redundant by our scheme. Table 9.2 also displays the effect of column-splitting process used in the second phase of two-phase approaches. In the table, $(M_c^{SB} - M_c^{DB})/N_\ell$ shows the average number of coupling rows induced by a linking column during the A_{DB}-to-A_{SB} transformation. It can easily be derived from the table that a linking column induces 1.27, 1.41 and 1.07 coupling rows in the FH, wBG and BG-onmetis schemes, respectively, on average. This means that the vertex separators found by these schemes contain column vertices with small connectivity, e.g., 2.27, 2.07 and 2.41.

TABLE 9.3
Performance of different techniques on the net-intersection-graph model.

	Indirect GPVS						Direct GPVS		
	NIG-model			wNIG-model			NIG-model		
	oemetis			*kmetis* + OR-MI			*onmetis*		
K	%M$_c$	%RI	%CI	%M$_c$	%RI	%CI	%M$_c$	%RI	%CI
4	5.12	6.7	9.0	2.60	9.9	11.2	1.57	7.7	12.7
8	7.67	13.4	16.9	4.76	15.2	18.8	3.31	15.3	23.3
16	10.27	22.6	30.0	6.72	18.6	25.2	5.46	24.8	35.5
Avg	7.69	14.2	18.6	4.69	14.6	18.4	3.44	16.0	23.8

Table 9.3 presents the results of our experiments on the NIG model for one-phase A-to-A_{SB} transformation. On the NIG model, we experimented with built-in GPVS tools in MeTiS (*oemetis* and *onmetis* in MeTiS, which are indirect and direct methods, respectively) and our weighted-NIG (wNIG) scheme. In our wNIG scheme, the edges of the net-intersection graph are weighted according to (5.1), and this edge-weighted NIG is provided to GPES tool *kmetis*. Then the resulting GPES partition is converted to a GPVS partition, using the proposed OR-MI heuristic for K-way wide-to-narrow separator refinement (OR-MI heuristic produced 2% smaller separators than MR and MI heuristics on average). Since the GPES and GPVS solvers of MeTiS maintain balance on vertices, the balance criterion on the row counts of the diagonal blocks were explicitly maintained during partitioning. The technique proposed in Section 7.2 need to be incorporated to the MeTiS package to explicitly maintain balance on the column counts of the diagonal blocks. As seen in Table 9.3, the proposed indirect wNIG scheme and *onmetis* produce substantially better SB forms than indirect *oemetis* scheme while wNIG scheme and *onmetis* are competitive, with *onmetis* having the edge. Since the indirect *oemetis* scheme utilizes unit edge weights, results support the validity of the edge weighting scheme for GPVS.

In our experiments with our hypergraph model, we have observed a significant sensitivity on the coarsening method employed. Although the coarsening algorithm might be considered as an implementation choice, significant affects on the performance motivated us for a brief discussion to give some insight to the potential users of the this work and for the reproducability of our experimental results. We should note that move-based local improvement heuristics like FM might fail miserably when used without the multilevel paradigm, due to very large net degrees, and a clever coarsening algorithm is crucial. Table 9.4 illustrates the performance comparison of two of the most effective clustering schemes with balance criteria used in our PaToH implementation. In the table, ABSM and ABSC denote the randomized matching-based and agglomerative clustering schemes used in the coarsening phase of PaToH. Both ABSM and ABSC schemes use the *absorption* metric [1] for vertex coalescing. Details of these clustering schemes and other implementation details can be found in PaToH manual [10]. For this work, we enhanced PaToH for maintaining different balance criteria. R-PaToH maintains balance on the number of internal nets of the parts during partitioning. (R&C)-PaToH maintains the balance on both the number of internal nets and vertices of the parts during partitioning. (R+C)-PaToH maintains balance on the sum of internal net and vertex counts of the parts during partitioning. Note that, in the row-net hypergraph model, balancing the internal net and vertex counts of the parts correspond, respectively, to balancing the row and column counts of the diagonal blocks of the resulting SB form.

As seen in Table 9.4, in R-PaToH, ABSC produces approximately 16% less number of coupling rows than ABSM, on average. We have observed a similar quality differ-

TABLE 9.4
Performance of different balancing criteria and clustering schemes in PaToH.

K	R-PaToH						(R&C)-PaToH			(R+C)-PaToH		
	ABSM			ABSC			ABSC			ABSC		
	%M$_c$	%RI	%CI	%M$_c$	%RI	%CI	%M$_c$	%RI	%CI	%M$_c$	%RI	%CI
4	1.95	8.9	13.9	1.62	9.1	15.0	1.72	8.2	10.1	1.69	10.1	10.2
8	3.61	15.9	25.9	3.15	15.6	26.3	3.43	14.5	17.2	3.31	16.7	16.6
16	5.78	23.2	35.3	4.79	23.5	37.1	5.17	21.3	24.6	4.98	25.6	23.9
Avg	3.78	16.0	25.0	3.19	16.1	26.2	3.44	14.7	17.3	3.33	17.4	16.9

ence between ABSC and ABSM both in (R&C)-PaToH and (R+C)-PaToH, therefore only ABSC results are displayed for the latter balancing schemes in the table. In the comparison of different balancing schemes using ABSC, R-PaToH performs better than (R+C)-PaToH, which performs better than (R&C)-PaToH in terms of the number of coupling rows. This observation can be explained by the decrease in the size of the feasible solution space with increasing difficulty of the balancing criterion.

Tables 9.5–9.7 illustrate the performance comparison of different schemes, all proposed (except FH) in this paper, on A-to-A_{SB} transformation. Table 9.5 and Table 9.6 display the quality of SB forms in terms of border size (%M_c) and diagonal-block imbalance (%RI and %CI), respectively, whereas Table 9.7 displays the the runtime performance of the algorithms. Since direct GPVS solvers perform better than indirect GPVS solvers as displayed in Tables 9.2 and 9.3, only the results of direct GPVS solver *onmetis* are given in Tables 9.5–9.7. Similarly, ABSC clustering results are presented for hypergraph partitionings with PaToH. The FH algorithm effectively maintains the balance on the sum of the row and column counts of the diagonal blocks. The proposed two-phase BG-onmetis scheme also works according to the same balance criterion, because of the limitation of the direct GPVS solver *onmetis*. Therefore for the sake of a common experimental framework, the results of (R+C)-PaToH, BG-onmetis, and FH schemes are grouped together in Tables 9.5–t7. Similarly, the results of one-phase R-PaToH and NIG-onmetis schemes are also grouped together.

As seen in Table 9.5, all of the proposed schemes perform significantly better than the FH algorithm. For example, the number of coupling rows of the SB forms produced by the FH algorithm are 3 times larger than those of the (R+C)-PaToH, on the overall average. In (R+C)-balancing, one-phase approach PaToH produces approximately 11% fewer coupling rows than two-phase approach BG-onmetis, on average, which confirms the effectiveness of the hypergraph model proposed for permuting rectangular matrices into SB forms. In R-balancing, PaToH produces approximately 7% fewer coupling rows than NIG-onmetis, on average. However, out of the 39 partitioning instances, NIG-onmetis performs better in 17 instances, whereas PaToH performs better in 18 instances. Recall that both schemes exploit the hypergraph model where NIG-onmetis scheme is based on the proposed formulation of the HP problem as a GPVS problem. As seen in the Table 9.5, the numbers of coupling rows of the SB forms produced by PaToH remain below 5/partitionings, in both (R+C) and R balancing schemes, on average. As seen in Tables 9.5–9.6, our methods can find balanced permutations, with very few coupling rows, which would lead to efficient parallel solutions.

Table 9.7 displays execution times of the partitioning algorithms. In two-phase approaches, hypergraph and bipartite representation of a rectangular matrix are of equal size: the number of nonzeros in the matrix. However, the clustering phase of an

Table 9.5
Overall performance comparison in A-to-A_{SB} transformation.

name	K	Balance on # of Rows + # of Cols (R+C)			Balance on # of Rows (R)	
		1-phase H-model	2-phase BG-model		1-phase H-model	
		PaToH	*onmetis*	FH	PaToH	*onmetis* NIG
		$\%M_c$	$\%M_c$	$\%M_c$	$\%M_c$	$\%M_c$
NL	4	5.02	5.22	27.71	5.02	5.16
	8	6.02	6.59	32.57	5.93	6.08
	16	7.19	8.31	36.72	7.14	7.49
CQ9	4	2.87	2.92	23.06	2.70	2.86
	8	4.10	4.03	27.76	3.90	3.84
	16	5.40	5.28	30.50	5.07	4.77
GE	4	3.01	2.53	4.71	3.00	2.78
	8	4.37	4.39	8.06	4.35	4.25
	16	5.63	5.97	10.81	5.54	5.68
CO9	4	2.72	2.78	21.27	2.67	2.37
	8	3.78	3.85	26.12	3.54	3.63
	16	5.10	5.03	30.26	4.85	4.96
car4	4	0.00	0.00	0.00	0.00	0.00
	8	0.00	0.52	1.29	0.00	0.00
	16	0.00	1.83	1.29	0.00	1.26
fxm4-6	4	0.64	0.41	0.49	0.46	0.44
	8	1.17	0.80	1.70	0.99	0.80
	16	2.13	1.42	2.28	1.90	1.41
fome12	4	0.00	0.00	0.00	0.00	0.00
	8	9.43	12.27	17.04	8.74	11.06
	16	15.39	21.23	29.02	14.75	19.08
pltexpA4-6	4	1.62	0.79	1.08	1.63	1.63
	8	3.02	2.15	1.98	2.94	2.84
	16	5.32	4.42	4.77	5.29	5.14
kent	4	0.34	0.15	0.66	0.28	0.15
	8	0.70	0.56	2.11	0.64	0.54
	16	1.26	1.33	3.47	1.08	1.10
world	4	1.08	0.80	1.53	0.86	0.75
	8	2.25	2.25	3.79	1.87	2.09
	16	5.25	5.94	9.29	4.76	5.61
mod2	4	0.86	0.78	0.88	0.76	0.65
	8	2.12	2.05	3.42	1.85	1.91
	16	5.10	5.64	8.75	4.72	5.46
lpl1	4	3.27	4.08	6.37	3.25	3.35
	8	5.40	6.58	9.03	5.62	5.67
	16	6.17	8.76	15.96	6.41	8.62
fxm3-16	4	0.52	0.33	0.56	0.42	0.26
	8	0.66	0.73	0.34	0.61	0.31
	16	0.86	1.51	0.39	0.75	0.33
Averages over K						
	4	1.69	1.60	6.80	1.62	1.57
	8	3.31	3.60	10.40	3.15	3.31
	16	4.98	5.90	14.12	4.79	5.46
	all	3.33	3.70	10.44	3.19	3.44

HP tool involves more costly operations than those of a GP tool. Hence, two-phase approaches using a GP tool is expected to run faster than the one-phase approach using an HP tool. In (R+C)-balancing, two-phase approach BG-onmetis runs faster than PaToH in the partitioning of all test matrices except *GE*, *car4* and *kent*. Recall that the NIG representation of the hypergraph model for a rectangular matrix A

Table 9.6
Balance quality in A-to-A_{SB} transformation.

		Balance on # of Rows + # of Columns (R+C)						Balance on # of Rows (R)			
		1-phase H-model		2-phase BG-model				1-phase H-model			
name	K	(R+C)-PaToH		*onmetis*		FH		R-PaToH		NIG-*onmetis*	
		%RI	%CI	%RI	%CI	%RI	%CI	%RI	%CI	%RI	%CI
NL	4	8.6	6.6	11.8	12.2	15.5	14.0	6.7	7.6	13.3	13.1
	8	13.0	11.3	17.6	18.5	23.4	19.2	10.0	11.4	19.6	18.0
	16	18.3	15.3	23.7	24.5	28.9	22.2	13.8	18.7	26.0	28.5
CQ9	4	17.0	22.6	17.8	17.6	19.5	17.8	6.5	42.8	13.4	43.0
	8	26.6	31.0	24.4	25.3	22.9	24.0	10.4	61.3	20.1	63.1
	16	37.3	38.6	36.7	29.5	29.5	24.8	16.6	77.1	25.8	75.5
GE	4	14.8	11.8	15.5	15.4	13.5	12.1	13.1	15.2	8.8	15.0
	8	21.5	19.8	19.3	20.0	19.0	19.4	17.2	29.0	18.9	25.2
	16	29.9	27.6	27.0	27.7	28.2	22.9	25.6	42.1	26.6	40.6
CO9	4	10.9	19.3	14.7	12.1	18.8	17.1	13.0	36.5	14.8	28.5
	8	14.4	27.5	21.2	16.9	20.7	24.4	18.8	57.2	18.1	36.5
	16	27.9	33.0	30.1	22.2	26.7	25.4	27.2	73.6	26.6	47.8
car4	4	0.6	0.9	3.3	5.9	22.6	25.0	0.0	2.6	0.0	2.6
	8	0.6	2.0	12.8	18.7	0.9	2.9	0.0	0.0	0.0	6.0
	16	0.7	4.3	23.7	36.1	0.9	6.1	0.0	12.9	11.0	21.5
fxm4-6	4	10.0	9.5	2.6	2.4	8.0	7.8	10.5	10.3	4.1	3.9
	8	14.7	13.8	10.9	11.0	14.8	14.3	17.1	16.7	10.8	10.5
	16	23.2	22.5	19.1	20.2	15.1	15.1	23.0	23.1	18.2	18.5
fome12	4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	8	9.6	8.3	12.8	10.7	16.6	13.3	9.5	14.7	21.6	16.8
	16	19.4	13.8	24.9	22.3	25.9	16.8	18.2	18.3	41.2	34.6
pltexpA4-6	4	5.9	4.8	2.9	3.2	11.4	11.7	6.4	5.7	7.0	5.7
	8	12.9	10.4	10.2	10.9	11.9	12.5	13.2	10.8	13.2	10.4
	16	19.7	17.0	16.2	18.2	15.5	16.7	19.0	17.0	21.7	18.7
kent	4	12.2	16.1	12.9	23.8	18.3	21.8	13.5	19.1	8.2	19.7
	8	19.3	24.6	21.3	35.0	22.9	18.8	16.0	35.8	13.9	36.8
	16	26.7	41.7	31.8	48.5	28.8	32.9	18.5	60.2	25.5	62.5
world	4	9.8	11.5	10.3	10.0	10.5	10.0	9.9	11.8	10.0	10.2
	8	17.8	20.6	17.8	19.8	15.4	17.5	22.3	25.9	20.9	25.1
	16	30.9	28.3	31.0	30.4	22.6	20.0	44.9	43.6	34.9	36.5
mod2	4	9.6	10.3	10.6	10.6	11.8	10.7	9.8	10.9	10.8	11.1
	8	17.2	18.3	18.4	20.4	15.0	17.1	21.4	23.9	21.8	25.3
	16	30.2	26.7	28.7	29.6	22.0	20.4	40.7	37.8	35.9	35.9
lpl1	4	18.4	6.8	11.7	5.5	13.4	12.3	16.1	20.8	9.3	11.8
	8	31.3	12.0	15.8	11.7	24.1	17.3	23.9	27.0	18.4	27.7
	16	40.5	16.0	26.0	18.9	35.8	20.3	30.2	30.9	25.8	38.5
fxm3-16	4	13.6	12.9	0.6	0.5	7.8	7.6	12.8	12.2	0.9	0.5
	8	17.6	16.6	1.3	1.2	2.4	1.8	22.6	21.9	1.9	1.2
	16	27.7	26.4	2.9	2.6	4.6	3.7	28.3	27.6	3.7	2.7
Averages over K											
	4	10.1	10.2	8.8	9.2	13.2	12.9	9.1	15.0	7.7	12.7
	8	16.7	16.6	15.7	16.9	16.2	15.6	15.6	26.3	15.3	23.3
	16	25.6	23.9	24.8	25.4	21.9	19.0	23.5	37.1	24.8	35.5
	all	17.4	16.9	16.4	17.2	17.1	15.8	16.1	26.2	16.0	23.8

corresponds to the standard graph representation of AA^T. Thus, relative sizes of the hypergraph and NIG models and hence relative execution times of GP and HP tools depend on the relative sizes of A and AA^T. Although NIG-onmetis runs faster than PaToH in general in R-balancing, PaToH runs faster in all partitionings of the *car4* and *kent* matrices. This is expected since *car4* and *kent* matrices have much less nonzeros than their AA^T matrices, as seen in Table 9.1.

TABLE 9.7
Execution times in seconds. Numbers in parentheses are normalized execution times with respect to (R+C)-PaToH

		Balance on # of Rows + # of Columns (R+C)			Balance on Balance on # of Rows (R)	
		1-phase (H-model)	2-phase (BG-model)		1-phase (H-model)	
name	K	(R+C)-PaToH	*onmetis*	FH [14]	R-PaToH	NIG-onmetis
NL	4	1.70 (1.00)	1.13 (0.67)	0.72 (0.42)	1.68 (0.99)	0.84 (0.49)
	8	1.97 (1.00)	1.44 (0.73)	0.72 (0.37)	2.00 (1.02)	0.97 (0.49)
	16	2.18 (1.00)	1.60 (0.73)	0.85 (0.39)	2.19 (1.00)	1.10 (0.50)
CQ9	4	2.55 (1.00)	1.88 (0.74)	1.18 (0.46)	2.58 (1.01)	1.33 (0.52)
	8	3.17 (1.00)	2.48 (0.78)	1.27 (0.40)	3.25 (1.03)	1.51 (0.48)
	16	3.72 (1.00)	2.98 (0.80)	1.46 (0.39)	3.84 (1.03)	1.79 (0.48)
GE	4	0.89 (1.00)	1.10 (1.24)	0.55 (0.62)	0.90 (1.01)	0.83 (0.93)
	8	1.19 (1.00)	1.56 (1.32)	0.62 (0.52)	1.23 (1.03)	1.08 (0.91)
	16	1.58 (1.00)	1.81 (1.15)	0.68 (0.43)	1.62 (1.03)	1.31 (0.83)
CO9	4	2.76 (1.00)	2.16 (0.78)	1.34 (0.48)	2.82 (1.02)	1.57 (0.57)
	8	3.43 (1.00)	2.78 (0.81)	1.45 (0.42)	3.50 (1.02)	1.90 (0.55)
	16	3.86 (1.00)	3.34 (0.87)	1.65 (0.43)	4.11 (1.07)	2.18 (0.57)
car4	4	1.99 (1.00)	25.64 (12.90)	26.00 (13.09)	2.07 (1.04)	4.46 (2.24)
	8	2.88 (1.00)	28.19 (9.78)	30.31 (10.51)	2.87 (0.99)	5.54 (1.92)
	16	3.74 (1.00)	29.25 (7.82)	32.54 (8.70)	3.72 (1.00)	6.22 (1.66)
fxm4-6	4	3.77 (1.00)	3.77 (1.00)	1.80 (0.48)	3.78 (1.00)	2.59 (0.69)
	8	5.61 (1.00)	5.59 (1.00)	1.86 (0.33)	5.67 (1.01)	3.53 (0.63)
	16	7.41 (1.00)	7.28 (0.98)	1.88 (0.25)	7.23 (0.98)	4.46 (0.60)
fome12	4	9.69 (1.00)	4.48 (0.46)	2.49 (0.26)	9.71 (1.00)	2.83 (0.29)
	8	15.06 (1.00)	8.82 (0.59)	3.22 (0.21)	15.10 (1.00)	5.98 (0.40)
	16	17.49 (1.00)	11.21 (0.64)	4.49 (0.26)	18.64 (1.07)	7.22 (0.41)
pltexpA4-6	4	4.37 (1.00)	4.08 (0.93)	2.17 (0.50)	4.47 (1.02)	2.29 (0.52)
	8	6.46 (1.00)	6.32 (0.98)	2.18 (0.34)	6.58 (1.02)	3.30 (0.51)
	16	8.41 (1.00)	7.80 (0.93)	2.25 (0.27)	8.59 (1.02)	4.23 (0.50)
kent	4	4.67 (1.00)	5.55 (1.19)	2.79 (0.60)	4.72 (1.01)	6.34 (1.36)
	8	6.90 (1.00)	8.24 (1.19)	3.01 (0.44)	7.00 (1.01)	9.18 (1.33)
	16	8.56 (1.00)	10.27 (1.20)	3.30 (0.39)	8.75 (1.02)	10.97 (1.28)
world	4	4.97 (1.00)	3.69 (0.74)	1.96 (0.39)	4.85 (0.98)	3.69 (0.74)
	8	7.11 (1.00)	5.56 (0.78)	2.07 (0.29)	7.18 (1.01)	5.27 (0.74)
	16	9.14 (1.00)	7.75 (0.85)	2.49 (0.27)	9.24 (1.01)	7.04 (0.77)
mod2	4	4.87 (1.00)	3.59 (0.74)	1.91 (0.39)	4.95 (1.02)	3.75 (0.77)
	8	6.80 (1.00)	5.48 (0.81)	2.00 (0.29)	7.08 (1.04)	5.45 (0.80)
	16	9.07 (1.00)	7.64 (0.84)	2.38 (0.26)	8.87 (0.98)	6.85 (0.76)
lpl1	4	31.65 (1.00)	12.95 (0.41)	6.42 (0.20)	30.66 (0.97)	5.73 (0.18)
	8	41.28 (1.00)	18.30 (0.44)	6.75 (0.16)	41.55 (1.01)	8.20 (0.20)
	16	46.39 (1.00)	25.16 (0.54)	7.51 (0.16)	44.59 (0.96)	10.01 (0.22)
fxm3-16	4	6.13 (1.00)	6.23 (1.02)	3.23 (0.53)	6.42 (1.05)	3.98 (0.65)
	8	9.37 (1.00)	9.10 (0.97)	3.10 (0.33)	9.48 (1.01)	5.38 (0.57)
	16	12.29 (1.00)	11.91 (0.97)	2.96 (0.24)	12.38 (1.01)	6.75 (0.55)
Averages over K						
	4	(1.00)	(1.76)	(1.42)	(1.01)	(0.77)
	8	(1.00)	(1.55)	(1.13)	(1.02)	(0.73)
	16	(1.00)	(1.41)	(0.96)	(1.01)	(0.70)
	all	(1.00)	(1.57)	(1.17)	(1.01)	(0.73)

Finally, we want to point that these runtimes are affordable when the solution times of the LP problems are considered. *LOQO* [62] solves the *lpl1* problem, which has the constraint matrix with the largest $M \times N$ product, in approximately 3800 seconds. As seen in Table 9.6, the 16-way partitioning times of all algorithms remain below 1.22% of the LOQO solution time in this LP problem.

10. Conclusion. We have investigated the problems of permuting a sparse rectangular matrix A into doubly-bordered (DB) and singly-bordered (SB) block-diagonal forms A_{DB} and A_{SB} with minimum border size while maintaining balance on the diagonal blocks. We showed that the A-to-A_{DB} transformation problem can be described as a graph-partitioning by vertex separator (GPVS) problem on the bipartite-graph representation of matrix A. We proposed a hypergraph model for representing the sparsity structure of A so that the A-to-A_{SB} transformation problem can be formulated as a hypergraph-partitioning (HP) problem. We proposed reducing the HP problem to the GPVS problem via net-intersection graph representation of a hypergraph. This reduction also shows that the A-to-A_{SB} transformation problem can be modeled as a GPVS problem on the standard graph representation of the symmetric matrix AA^T. The performance of the proposed models and approaches depends on the performance of the tools used to solve the associated problems as well as the representation power of the models. We also presented a brief survey on the solution techniques and tools for solving the stated problems, developed in different societies. Our experiments covered various techniques for each of the partitioning problems through using state-of-the-art multilevel GP and HP tools MeTiS and PaToH.

We have also investigated indirect GPVS approaches. These approaches perform a graph partitioning by edge separator (GPES) and take the boundary vertices as the wide vertex separator to be refined to a narrow separator. We proposed a new edge-weighting model so that minimizing the weighted edge cut through a GPES tool is expected to produce a wide vertex separator with better quality for refinement. Experimental results showed that the proposed model leads to substantially smaller narrow separators than the conventional indirect GPVS solver *oemetis* while approaching to the quality of the direct GPVS solver *onmetis*. The proposed edge-weighting model can also be exploited in the coarsening phase of the multilevel direct GPVS solvers. We also presented a short discussion on the optimality of the wide-to-narrow separator refinement through minimum vertex cover model.

We explored the effectiveness of the proposed techniques on transforming general Linear Programming (LP) problems into block-angular forms through permuting their constraint matrices into SB forms. Experimental results on a large collection of LP problems were impressive, and verified that the underlying block-angular structures of the LP problems can be effectively extracted to enable the use of the decomposition-based techniques for coarse-grain parallel solution of general LP problems.

REFERENCES

[1] C.J. Alpert and A.B. Kahng, "Recent Directions in Netlist Partitioning: A survey," *VLSI Journal*, vol. 19, nos. 1-2, pp. 1–81, 1995.

[2] C. Ashcraft and J.W.H. Liu, "A Partition Improvement Algorithm for Generalized Nested Dissection," Technical Report, BCSTECH-94-020, Boeing Computer Services, Seattle, WA, 1994.

[3] C. Ashcraft and J.W.H. Liu, "Applications of the Dulmage-Mendelsohn Decomposition and Network Flow to Graph Bisection Improvement," *SIAM Journal on Matrix Analysis and Applications*, vol. 19, no. 2, pp. 325–354, 1998.

[4] Ake Bjorck *Numerical Methods for Least Squares Problems*, SIAM Press, 1996.

[5] C. Aykanat, Ali Pınar and Ü.V. Çatalyürek, "Experiments with Models and Heuristics for Permuting Sparse Rectangular Matrices into Block-Diagonal Form," Technical Report BU–CEIS–200202, Computer Engineering Department, Bilkent University, Turkey, 2002.

[6] T. Bui and C. Jones, "Finding Good Approximate Vertex and Edge Partitions is NP-hard," *Information Processing Letters*, vol. 42, pp. 153–159, 1992.

[7] T. Bui and C. Jones, "A Heuristic for Reducing Fill in Sparse Matrix Factorization," *Proceedings of 6th SIAM Conf. Parallel Processing for Scientific Computing*, pp. 445–452, 1993.

[8] Ü.V. Çatalyürek and C. Aykanat, "Decomposing Irregularly Sparse Matrices for Parallel Matrix-Vector Multiplications," *Proceedings of 3rd International Symposium on Solving Irregularly Structured Problems in Parallel, Irregular'96 (Lecture Notes in Computer Science, vol. 1117)*, pp. 75–86, 1996.

[9] Ü.V. Çatalyürek and C. Aykanat, "Hypergraph-Partitioning-Based Decomposition for Parallel Sparse-Matrix Vector Multiplications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 10. no. 7, July 1999.

[10] Ü.V. Çatalyürek and C. Aykanat, "PaToH a Multilevel Hypergraph Partitioning Tool for Decomposing Sparse Matrices and Partitioning VLSI Circuits," Technical Report BU–CEIS–9902, Department of Computer Engineering and Information Science, Bilkent University, Turkey, 1999.

[11] J. Cong, L. Hagen, and A.B. Kahng, "Net Partitions Yield Better Module Partitions," *Proceedings of 29th ACM/IEEE Design Automation Conference*, pp. 47–52, 1992.

[12] J. Cong, W. Labio, and N. Shivakumar, "Multi-Way VLSI Circuit Partitioning Based on Dual Net Representation," *Proceedings of IEEE International Conference on Computer-Aided Design*, pp. 56–62, 1994.

[13] G.B. Dantzig and P. Wolfe, "Decomposition Principle for Linear Programs," *Operations Research*, vol. 8, pp. 101–111, 1960.

[14] M.C. Ferris and J. D. Horn, "Partitioning Mathematical Programs for Parallel Solution," *Mathematical Programming*, vol. 80, pp. 35–61, 1988.

[15] C.M. Fiduccia and R.M. Mattheyses, "A Linear-Time Heuristic for Improving Network Partitions," *Proceedings of 19th ACM/IEEE Design Automation Conference*, pp. 175–181, 1982.

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

[17] D.M. Gay, "Electronic mail distribution of linear programming test problems," *Mathematical Programming Society COAL Newsletter*, 1985.

[18] A. George, "Nested dissection of a regular finite element mesh," *SIAM Journal on Numerical Analysis*, vol. 10, pp. 14–16, 1973.

[19] A. George and J.W.H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice–Hall, Englewood Cliffs, NJ, 1981.

[20] S.K. Gnanendran and J.K. Ho, "Load Balancing in the Parallel Optimization of Block-Angular Linear Programs," *Mathematical Programming*, vol. 62, pp. 41–67, 1993.

[21] A. Gupta, "Fast and Effective Algorithms for Graph Partitioning and Sparse Matrix Ordering," Technical Report RC 20496, IBM T. J. Watson Research Center, Yorktown Heights, NY, 1996.

[22] A. Gupta, "Watson Graph Partitioning Package," Technical Report RC 20453, IBM T. J. Watson Research Center, Yorktown Heights, NY, 1996.

[23] G. Hachtel, A.R. Newton, and A. Sangiovanni-Vincentelli, "An Algorithm for Optimal PLA Folding," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 1, no. 2, pp. 63–77, 1982.

[24] S. Hauck and G. Boriello, "An Evaluation of Bipartitioning Techniques," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 8, pp. 849–866, 1997.

[25] B. Hendrickson and R. Leland, "A Multilevel Algorithm for Partitioning Graphs," Technical Report, Sandia National Laboratories, 1993.

[26] B. Hendrickson and R. Leland, "The Chaco User's Guide, Version 2.0," Technical Report SAND95-2344, Sandia National Laboratories, Albuquerque, NM, 87185, 1995.

[27] B. Hendrickson, "Graph Partitioning and Parallel Solvers: Has the Emperor no Clothes?," *Proceedings of Fifth International Symposium on Solving Irregularly Structured Problems in Parallel, Irregular'98 (Lecture Notes in Computer Science, vol. 1457)*, pp. 218–225, 1998.

[28] B. Hendrickson and E. Rothberg, "Improving the Run Time and Quality of Nested Dissection Ordering," *SIAM Journal on Scientific Computing*, vol. 20, no. 2, pp. 468–489, 1998.

[29] B. Hendrickson and T.G. Kolda, "Partitioning Rectangular and Structurally Nonsymmetric Sparse Matrices for Parallel Processing," *SIAM Journal on Scientific Computing*, vol. 21, no. 6, pp. 2048–2072, 2000.

[30] B. Hendrickson and T.G. Kolda, "Graph Partitioning Models for Parallel Computing," *Parallel Computing*, vol. 26, no. 12, pp. 1519–1534, 2000.

[31] J.K. Ho, T.C. Lee, and R.P. Sundarraj, "Decomposition of Linear Programs Using Parallel Computation," *Mathematical Programming*, vol. 42, pp. 391–405, 1998.

[32] Y. F. Hu and C. M. Maguire and R. J. Blake, "Ordering Unsymmetric Matrices into Bordered

Block Diagonal Form for Parallel Processing,"

[33] Hungarian Academy of Sciences: Computer and Automation Research Institute, LP Test Sets, ftp://ftp.sztaki.hu/pub/oplab/LPTESTSET/.

[34] E. Ihler, D. Wagner, and F. Wagner, Modeling Hypergraphs by Graphs with the Same Mincut Properties, *Information Processing Letters*, vol. 45, pp. 171–175, 1993.

[35] "IMSL User's Manual, Edition 9.2 (International Mathematical and Statistical Library)," Houston, TX, Nov. 1984.

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

[37] A.B. Kahng, "Fast Hypergraph Partitioning," *Proceedings of 26th ACM/IEEE Design Automation Conference*, pp. 762–766, 1989.

[38] G. Karypis and V. Kumar, "MeTiS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices Version 4.0," Technical Report, University of Minnesota, Department of Computer Science and Engineering, Army HPC Research Center, Minneapolis, 1998.

[39] 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.

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

[41] G. Karypis and V. Kumar, "Multilevel k-way Partitioning Scheme for Irregular Graphs," *Journal of Parallel and Distributed Computing*, vol. 48, pp. 86–129, 1998.

[42] G. Karypis and V. Kumar, "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs," *SIAM Journal on Scientific Computing*, to appear.

[43] B.W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," *The Bell System Technical Journal*, vol. 29, pp. 291–307, 1970.

[44] T.G. Kolda, "Partitioning Sparse Rectangular Matrices for Parallel Processing," *Proceedings of 5th International Symposium on Solving Irregularly Structured Problems in Parallel, Irregular'98 (Lecture Notes in Computer Science, vol. 1457)*, pp. 473–482, 1998.

[45] E.L. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart, and Winston, 1976.

[46] C.E. Leiserson and J. G. Lewis, "Orderings for Parallel Sparse Symmetric Matrix Factorization," *Third SIAM Conference on Parallel Processing for Scientific Computing*, pp. 27–31, 1987.

[47] C. Lemarechal, A. Nemirovski, and Y. Nesterov, "New Variants of Bundle Methods," *Mathematical Programming*, vol. 69, pp. 111–147, 1995.

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

[49] J.W.H. Liu, "A Graph Partitioning Algorithm by Node Separators," *ACM Transactions on Mathematical Software*, vol. 15, pp. 141–153, 1989.

[50] D. Medhi, "Parallel Bundle-Based Decomposition for Large-Scale Structured Mathematical Programming Problems," *Annals of Operations Research*, vol. 22, pp. 101–127, 1990.

[51] D. Medhi, "Bundle-Based Decomposition for Structured Large-Scale Convex Optimization: Error Estimate and Application to Block-Angular Linear Programs," *Mathematical Programming*, vol. 66, pp. 79–101, 1994.

[52] R. R. Meyer and G. Zakeri "Multicoordination Methods for Solving Convex Block-Angular Programs," *SIAM Journal on Optimization*, vol. 10, no. 1, pp. 121–131, 1999.

[53] J.M. Mulvey and A. Ruszcynski, "A Diagonal Quadratic Approximation Method for Large Scale Linear Programs," *Operations Research Letters*, vol. 12, pp. 205–215, 1992.

[54] S.S. Nielsen and S.A. Zenios, "A Massively Parallel Algorithm for Nonlinear Stochastic Network Problems," *Operations Research*, vol. 41, no. 2, pp. 319–337, 1993.

[55] A. Pınar, Ü.V. Çatalyürek, C. Aykanat, and M. Pınar, "Decomposing Linear Programs for Parallel Solution," *Proceedings of the Second International Workshop on Applied Parallel Computing, PARA'95 (Lecture Notes in Computer Science, vol. 1041)*, pp. 473–482, 1996.

[56] A. Pınar and C. Aykanat, "An Effective Model to Decompose Linear Programs for Parallel Solution," *Proceedings of the Third International Workshop on Applied Parallel Computing, PARA'96 (Lecture Notes in Computer Science, vol. 1184)*, pp. 592–601, 1997.

[57] A. Pothen and C.-J. Fan, "Computing the Block Triangular Form of a Sparse Matrix," *ACM Transactions on Mathematical Software*, vol. 16, no. 4, pp. 303–324, 1990.

[58] A. Pothen, H. D. Simon, and K.-P. Liou, "Partitioning Sparse Matrices with Eigenvectors of Graphs," *SIAM Journal on Matrix Analysis and Applications*, vol. 10, no. 3, pp. 430–452,

1990.

[59] T. Rashid and T. Davis, "An Approach for Parallelizing any General Unsymmetric Sparse Matrix Algorithm," *Proceedings of the Seventh SIAM Conf. on Parallel Processing for Scientific Computing*, pp. 413–417, Feb. 1995.

[60] S.M. Robinson, "Bundle-Based Decomposition: Description and Preliminary Results," *Lecture Notes in Control and Information Science*, vol. 84, pp. 751–756, 1986.

[61] J.M.Stern and S.A.Vavasis "Active Set Algorithms for Problems in Block Angular Form. Comp. Appl. Math.," vol. 12, no.3, pp. 199-226, 1994.

[62] R.J. Vanderbei, "*LOQO* User's Manual, Version 4.01," Technical Report SOR 97-08, Princeton University, 1997.