SIAM J. SCI. COMPUT.
Vol. 25, No. 6, pp. 1860–1879

PERMUTING SPARSE RECTANGULAR MATRICES INTO BLOCK-DIAGONAL FORM*

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

Abstract. We investigate the problem of permuting a sparse rectangular matrix into block-diagonal form. Block-diagonal form of a matrix grants an inherent parallelism for solving the deriving problem, as recently investigated in the context of mathematical programming, LU factorization, and QR factorization. To represent the nonzero structure of a matrix, we propose bipartite graph and hypergraph models that reduce the permutation problem to those of graph partitioning by vertex separator and hypergraph partitioning, respectively. Our experiments on a wide range of matrices, using the 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. 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

DOI. 10.1137/S1064827502401953

1. Introduction.

Block-diagonal structure of sparse matrices has been exploited for coarse-grain parallelization of various algorithms such as decomposition methods for linear programming, LU factorization, and QR factorization. In these methods, block diagonals give rise to subproblems that can be solved independently, whereas the border incurs a coordination task to combine the subproblem solutions into a solution of the original problem and is usually less amenable to 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.

Our target problem is permuting rows and columns of an $M \times N$ sparse matrix A into a K-way singly bordered block-diagonal (SB) form:

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

where P and Q denote, respectively, the row and column permutation matrices to be determined. In (1.1), each row of the $M_c \times N$ border submatrix $R = (R_1 \ \cdots \ R_K)$

*Received by the editors February 4, 2002; accepted for publication (in revised form) August 25, 2003; published electronically May 25, 2004.
http://www.siam.org/journals/sisc/25-6/40195.html

†Computer Engineering Department, Bilkent University, Ankara, Turkey (aykanat@cs.bilkent.edu.tr). This author's work was partially supported by the Scientific and Technical Research Council of Turkey (TÜBİTAK) under project EEEAG-199E013.

‡Lawrence Berkeley Laboratory, Berkeley, CA 94720 (apinar@lbl.gov). This author's work was supported by the Director, Office of Science, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy under contract DE-AC03-76SF00098.

§Department of Biomedical Informatics, Ohio State University, Columbus, OH 43210 (catalyurek.1@osu.edu). This author's work was supported by the National Science Foundation under grant ACI-0203846.

is called a *column-coupling* or simply *coupling* row. Each coupling row has nonzeros in the columns of at least two diagonal blocks. 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 is satisfied. The SB form in (1.1) is referred to here as the *primal* SB form, whereas in the *dual* SB form they are the columns that constitute the border. We also consider the problem of permuting rows and columns of a sparse matrix A into a K-way doubly bordered block-diagonal (DB) form:

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

In equation (1.2), 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 sum of the number of coupling rows and columns is minimized while a given balance criterion is satisfied.

The literature that addresses this problem is very rare and recent. Ferris and Horn [12] 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 [38, 39]. In [38], 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 [39] we proposed our graph models. Later Hu, Maguire, and Blake [24] 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 forms.

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 adoption of algorithms and tools for these well-studied problems to permute sparse matrices to block-diagonal form efficiently and effectively.

In this work, we show that the A-to-A_{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 columns in A_{DB}.

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 [38]—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}.

The organization of the paper is as follows: In the next section we will discuss how block-diagonal structure can be exploited in parallelization of various applications. Some preliminary information on graph and hypergraph partitioning and A_{DB}-to-A_{SB} transformation are presented in section 3. Our proposed models for A-to-A_{DB} and A-to-A_{SB} transformations are explained in sections 4 and 5, respectively. Section 6 overviews recent graph and hypergraph partitioning algorithms and tools. Experimental evaluation of the proposed models is presented in section 7. And finally section 8 concludes the paper.

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, and LU and QR factorizations.

2.1. Linear programming. Exploiting the block-angular structure of linear programs (LPs) dates back to the work of Dantzig and Wolfe [11], when the motivation was solving large LPs with limited memory. Later studies investigated parallelization techniques [15, 23, 34]. The proposed techniques [11, 31, 35] led 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 LPs 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 the manager-worker scheme [12, 15, 23, 34]. 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 proposed in [12], 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 the 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 [12, 34]. 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 balancing in the parallel phase.

It is worth noting that exploiting the block-angular structure of the constraint matrices is not restricted to LPs and can be applied in different optimization problems [36, 42].

2.2. LU factorization. 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 [24, 41] 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 $A_k^\pi = B_k$ after the unfactored rows/columns are permuted to the end of the matrix. In a subsequent phase, the coupling rows and columns, along with 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. Our first objective is to minimize 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.

2.3. QR factorization. Least squares is one of the fundamental problems in numerical linear algebra and is 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,

$$\begin{bmatrix} B_1 & & & C_1 \\ & B_2 & & C_2 \\ & & \ddots & \vdots \\ & & B_K & C_K \end{bmatrix},$$

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

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

In a subsequent phase, we factor $C' = \begin{bmatrix} C'_1, \ldots, C'_K \end{bmatrix}^T$ [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. In this section we will provide the basic definitions and techniques that will be adopted in the remainder of this paper.

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.

The objective of graph partitioning is finding a separator, whose removal decomposes the graph into disconnected subgraphs with balanced sizes. The separator can be a set of edges or a set of vertices, and associated problems are called graph partitioning by edge separator (GPES) and graph partitioning by vertex separator (GPVS) problems, respectively. Both GPES and GPVS problems are known to be NP-hard [5]. Balance among subgraphs is usually defined by cumulative effect of weights assigned to vertices. Some alternatives have been studied recently [40]. We proceed with formal definitions. $\Pi_{ES} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K\}$ 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 $1 \leq k < \ell \leq K$; $\bigcup_{k=1}^{K} \mathcal{V}_k = \mathcal{V}$. Edges between the vertices of different parts belong to $\mathcal{E}_S$ and are called *cut (external)* edges, and all other edges are called *uncut (internal)* edges.

DEFINITION 3.1 (GPES problem). *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, an integer K, and a balance criterion for subgraphs, the GPES problem is finding a K-way vertex partition $\Pi_{ES} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K\}$ of $\mathcal{G}$ by edge separator $\mathcal{E}_S$ that satisfies the balance criterion with minimum cost. The cost is defined as*

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

where w_{ij} is the weight of edge $e_{ij} = (v_i, v_j)$.

The GPVS problem is similar, except that a subset of vertices, as opposed to edges, serve as the separator. $\Pi_{VS} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K; \mathcal{V}_S\}$ 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 $1 \leq k \leq K$; $\mathcal{V}_k \cap \mathcal{V}_\ell = \emptyset$ for $1 \leq k < \ell \leq K$ and $\mathcal{V}_k \cap \mathcal{V}_S = \emptyset$ for $1 \leq k < K$; $\bigcup_{k=1}^{K} \mathcal{V}_k \cup \mathcal{V}_S = \mathcal{V}$; 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 $1 \leq k \leq K$). 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.

DEFINITION 3.2 (GPVS problem). *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, an integer K, and a balance criterion for subgraphs, the GPVS problem is finding a K-way vertex separator $\Pi_{VS} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K; \mathcal{V}_S\}$ that satisfies the balance criterion, with minimum cost, where the cost is defined as*

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

The techniques for solving GPES and GPVS problems are closely related, as will be further discussed in section 6. An *indirect* approach to solving the GPVS problem is to first find an edge separator through GPES, and then translate it to a vertex separator. After finding an edge separator, this approach takes vertices adjacent to separator edges as a wide separator to be refined to a narrow separator, with the assumption that a small edge separator yields a small vertex separator. The approach adopted by Ferris and Horn [12] falls into this class. The wide-to-narrow refinement problem is described as a *minimum vertex cover* problem on the bipartite graph induced by the cut edges. A minimum vertex cover can be taken as a narrow separator for the whole graph, because each cut edge will be adjacent to a vertex in the vertex cover.

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 denotes 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.

$\Pi_{HP} = \{\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_K\}$ 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 $\mathcal{N}_1, \ldots, \mathcal{N}_K$.

DEFINITION 3.3 (HP problem). *Given a hypergraph $\mathcal{H} = (\mathcal{U}, \mathcal{N})$, an integer K, and a balance criterion for subhypergraphs, the HP problem is finding a K-way partitioning $\Pi_{HP} = \{\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_K\}$ of $\mathcal{H}$ that satisfies the balance criterion, and minimizes the cost, which is defined as*

$$cost(\Pi_{HP}) = |\mathcal{N}_S| . \tag{3.3}$$

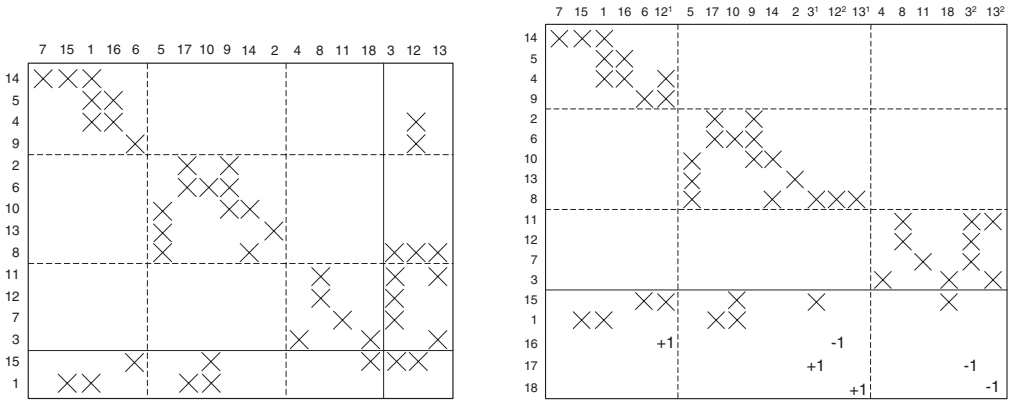


FIG. 3.1. *Column-splitting process.*

The above metric of cost is often referred to as the *cutsize* metric in VLSI community. The connectivity metric is defined as

$$cost(\Pi_{HP}) = \sum_{n_i \in \mathcal{N}_S} (\lambda_i - 1) \tag{3.4}$$

and is frequently used in VLSI [32] and scientific computing communities [8].

3.3. Column-splitting method for A_{DB}-to-A_{SB} transformation. In the second phase of the Ferris–Horn (FH) algorithm [12], A_{DB} is transformed into an SB form through the *column-splitting* technique used in stochastic programming to treat anticipativity [37]. In this technique, we consider the variables corresponding to the coupling columns. Consider a coupling 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 coupling 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 $|\Lambda_j| - 1$. Figure 3.1 depicts the column-splitting process on the A_{DB} matrix obtained in Figure 4.2b.

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 $\mathcal{B}_A = (\mathcal{V}, \mathcal{E})$ 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. Figure 4.2a displays the bipartite graph representation of the sample matrix given in Figure 4.1.

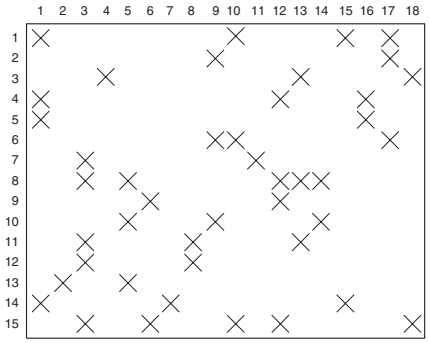


FIG. 4.1. *A* 15×18 *sample matrix A.*

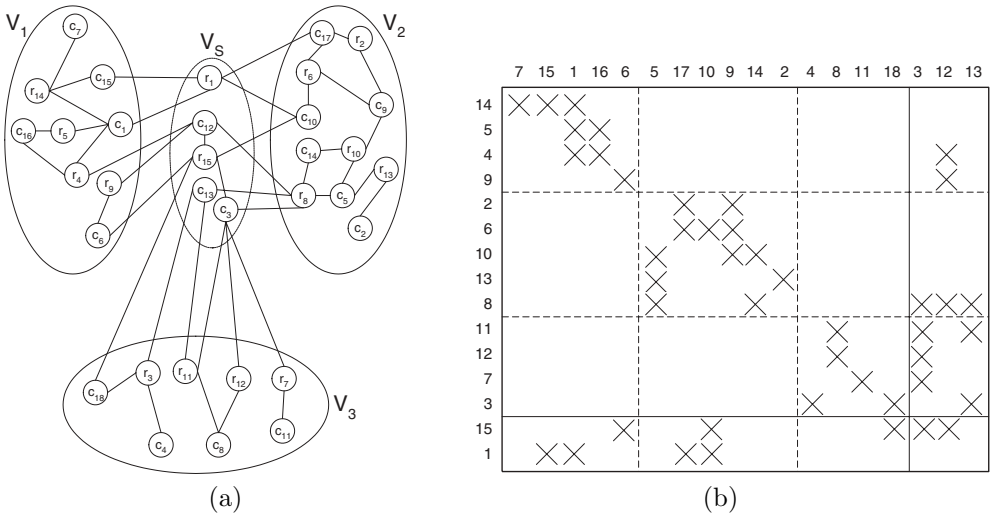


FIG. 4.2. (a) *Bipartite graph representation* $\mathcal{B}_A$ *of the sample A matrix given in Figure* 4.1 *and* 3*-way partitioning* Π_{VS} *of* $\mathcal{B}_A$ *by vertex separator;* (b) 3*-way DB form* A_{DB} *of A induced by* Π_{VS}.

Consider a K-way partition $\Pi_{VS} = \{\mathcal{V}_1, \ldots, \mathcal{V}_K; \mathcal{V}_S\}$ 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_{VS} 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 $k = 1, \ldots, K-1$, and the rows and columns associated with the vertices in $\mathcal{R}_S$ and $\mathcal{C}_S$ are ordered last as the coupling rows and columns, respectively.

THEOREM 4.1. *Let* $\mathcal{B}_A = (\mathcal{V}, \mathcal{E})$ *be the bipartite graph representation of a given matrix A. A* K*-way vertex separator* $\Pi_{VS} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K; \mathcal{V}_S\}$ *of* $\mathcal{B}_A$ *gives a permutation of A to* K*-way DB form* A_{DB}*, where row and column vertices in* $\mathcal{V}_k$ *constitute the rows and columns of the* k*th diagonal block of* A_{DB}*, and row and column vertices in* $\mathcal{V}_S$ *constitute the coupling rows and columns of* A_{DB}*. Thus,*

- *minimizing the size of the separator minimizes the border size;*
- *balance among subgraphs infer balance among diagonal submatrices.*

Proof. Consider a row vertex $r_i \in \mathcal{R}_k$ and a column vertex $c_j \in \mathcal{C}_k$ of part $\mathcal{V}_k$ in a partition Π_{VS} 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

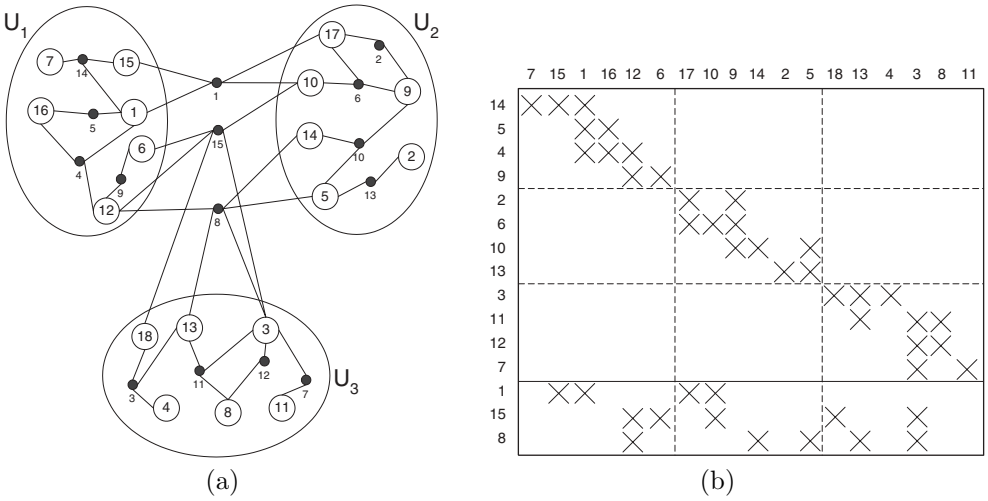
all nonzeros of row i of A into either submatrix A^{π}_{kk} or submatrices A^{π}_{kk} and A^{π}_{kS} depending on r_i being a nonboundary or a boundary vertex of $\mathcal{V}_k$, respectively. So, all nonzeros in the kth row slice A^{π}_{k*} of A^{π} will be confined to the A^{π}_{kk} and A^{π}_{kS} 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 nonboundary 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 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 $N_c = |\mathcal{C}_S|$. 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 columns in A^{π}, since $|\mathcal{V}_S| = |\mathcal{R}_S| + |\mathcal{C}_S| = M_c + N_c$. 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}. Figure 4.2a displays a 3-way GPVS of $\mathcal{B}_A$, and Figure 4.2b shows a corresponding partial permutation that transforms matrix A of Figure 4.1 into a 3-way DB form A_{DB}. □

5. 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 studies [7, 8, 38, 39], 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: the row-net representation of a matrix is equal to the column-net representation of its transpose. Here, we describe and discuss only 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 form 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, $Pins(n_i)$ represents the set of columns that have a nonzero in row i of 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. Figure 5.1a displays the hypergraph representation of the 16×18 sample matrix in Figure 4.1.

Recently, we exploited the proposed row-net (column-net) model for columnwise (rowwise) decomposition of sparse matrices for parallel matrix-vector multiplication [7, 8]. In that application, nodes represent units of computation and nets encode multiway data dependencies. In [7, 8], we showed that a one-dimensional matrix partitioning problem can be modeled as an HP problem in which the connectivity



(a) (b)

FIG. 5.1. (a) *Row-net hypergraph representation* $\mathcal{H}_A$ *of the sample* A *matrix shown in Figure* 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}*.*

metric in (3.4) is exactly equal to the parallel communication volume. The proposed HP model overcomes some flaws and limitations of the standard GPES models, which are also addressed by Hendrickson and Kolda [18, 19]. 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 in (3.3) is exactly equal to the number of coupling rows in A_{SB}.

THEOREM 5.1. *Let* $\mathcal{H}_A = (\mathcal{U}, \mathcal{N})$ *be the hypergraph representation of a given matrix* A*. A* K*-way partition* $\Pi_{HP} = \{\mathcal{U}_1, \ldots, \mathcal{U}_K\} = \{\mathcal{N}_1, \ldots, \mathcal{N}_K; \mathcal{N}_S\}$ *of* $\mathcal{H}_A$ *gives a permutation of* A *to* K*-way SB form, where nodes in* $\mathcal{U}_k$ *and internals nets in* $\mathcal{N}_k$*, respectively, constitute the columns and rows of the* k*th diagonal block of* A_{SB}*, and external nets in* $\mathcal{N}_S$ *constitute the coupling rows of* A_{SB}*. Thus,*

- *minimizing the cutsize minimizes the number of coupling rows;*
- *balance among subhypergraphs infer balance among diagonal submatrices.*

Proof. 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 $A^\pi_{*k} = \left((A^\pi_{1k})^T \cdots (A^\pi_{Kk})^T \, (A^\pi_{Sk})^T\right)^T$ of A^π. An internal net n_i of $\mathcal{U}_k$ corresponds to permuting row i of A to the kth row slice $A^\pi_{k*} = (A^\pi_{k1} \ \cdots \ \mathcal{A}^\pi_{kK})$ of A^π, and an external net n_i corresponds to permuting row i of A to the border $A^\pi_S = (A^\pi_{S1} \ \cdots \ \mathcal{A}^\pi_{SK})$ of A^π.

Consider an internal net $n_i \in \mathcal{N}_k$ of part $\mathcal{U}_k$ in a partition Π_{HP} 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 whether u_j is a nonboundary 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) 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}. Figure 5.1a displays a 3-way partitioning Π_{HP} of $\mathcal{H}_A$ and Figure 5.1b shows a corresponding partial permutation which transforms matrix A in Figure 4.1 directly into a 3-way SB form. □

6. Graph and hypergraph partitioning algorithms and tools. Recently, *multilevel* GPES [6, 20] and HP [8, 17, 29] approaches have been proposed, leading to successful GPES tools such as Chaco [21], MeTiS [27], and WGPP [16] and HP tools hMeTiS [29] and PaToH [9]. 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 decreases 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 [13] as a faster implementation of the KL algorithm proposed by Kernighan and Lin [30]. 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 the direct K-way GPES version of MeTiS [28] (*kmetis* option [27]) for indirect GPVS in the A-to-A_{DB} transformation phase of the FH method and our multilevel HP tool PaToH [9] in our one-phase A-to-A_{SB} transformation approach.

One of the most important applications of GPVS is George's *nested-dissection* algorithm [14], which has been widely used in fill-reducing orderings for sparse matrix factorizations. The basic idea in the nested-dissection algorithm is to reorder a 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 a K-way DB form of matrix A in our two-phase approach by providing the bipartite graph model of A as input to a nested-dissection-based reordering tool. Note that we effectively perform a nonsymmetric nested dissection on the bipartite graph model of the rectangular A matrix.

Direct 2-way GPVS approaches have been embedded into various multilevel nested-dissection implementations [16, 22, 27]. 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 [1]. This scheme considers vertex moves from vertex separator $\mathcal{V}_S$ to both $\mathcal{V}_1$ and $\mathcal{V}_2$ in $\Pi_{VS} = \{\mathcal{V}_1, \mathcal{V}_2; \mathcal{V}_S\}$. This refinement scheme is adopted in the *onmetis* ordering code of MeTiS [27], the ordering code of WGPP [16], and the ordering code BEND [22]. The second scheme is based on Liu's narrow separator refinement algorithm [33], which considers moving a set of vertices simultaneously from $\mathcal{V}_S$, in contrast to the FM-based refinement scheme [1], which moves only one vertex at a time. Liu's refinement algorithm [33] 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 between $\mathcal{V}_2$ 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 [2], and Hendrickson and Rothberg [22] enabled the use of Liu's refinement approach [33] on the coarse graphs within the multilevel framework. In this work, we use the publicly available *onmetis* ordering code of MeTiS [27] for direct GPVS.

7. Experimental results. We tested the performance of the proposed models and associated solution approaches on a wide range of large LP constraint matrices obtained from [10] and [25]. Properties of these rectangular matrices are presented in Table 7.1, where the matrices are listed in the order of increasing number of rows.

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 [27] and PaToH [9] were run 50 times starting from different random seeds for each instance. We use averages of these runs for each instance in this section. Figure 7.1 displays K = 4-, 8-, and 16-way sample primal SB forms of the matrix GE obtained by PaToH.

In this section, we first compare different solution techniques for a model. Tables 7.2–7.3 present only the averages over the 13 matrices. Breakdown of the results for each matrix can be found in [3]. Then we compare the effectiveness of the models for their best solution technique, both in terms of solution quality (Tables 7.4–7.5) and preprocessing times (Table 7.6). In these tables, $\%M_c$ denotes the percentage of the number of coupling rows in both DB and primal SB forms, i.e., $\%M_c = 100 \times M_c/M$. $\%N_c$ denotes the number of coupling columns in the DB forms as percents of the respective M values to enable the comparison of the M_c and N_c values under the same unit, i.e., $\%N_c = 100 \times N_c/M$. 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 $N_{avg} = (N - N_c)/K$ 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 $T(M, N) = cM^{2.17}N^{0.89}$ (where c is some constant) was reported in [34] for the solution time (with IMSL routine ZX0LP [26]) of an LP subproblem corresponding to an $M \times N$ diagonal block.

TABLE 7.1
Properties of rectangular test matrices.

Name	Number of rows M	Number of cols N	Number of nonzeros Total	per row max	per row avg	per col max	per col avg
NL	7039	9718	41428	149	5.89	15	4.26
CQ9	9278	13778	88897	390	9.58	24	6.45
GE	10099	11098	39554	47	3.92	36	3.56
CO9	10789	14851	101578	440	9.41	28	6.84
car4	16384	33052	63724	111	3.89	109	1.93
fxm4-6	22400	30732	248989	57	11.12	24	8.10
fome12	24284	48920	142528	228	5.87	14	2.91
pltexpA4-6	26894	70364	143059	30	5.32	8	2.03
kent	31300	16620	184710	960	5.90	18	11.11
world	34506	32734	164470	341	4.77	16	5.02
mod2	34774	31728	165129	310	4.75	16	5.20
lpl1	39951	125000	381259	177	9.54	16	3.05
fxm3-16	41340	64162	370839	57	8.97	36	5.78

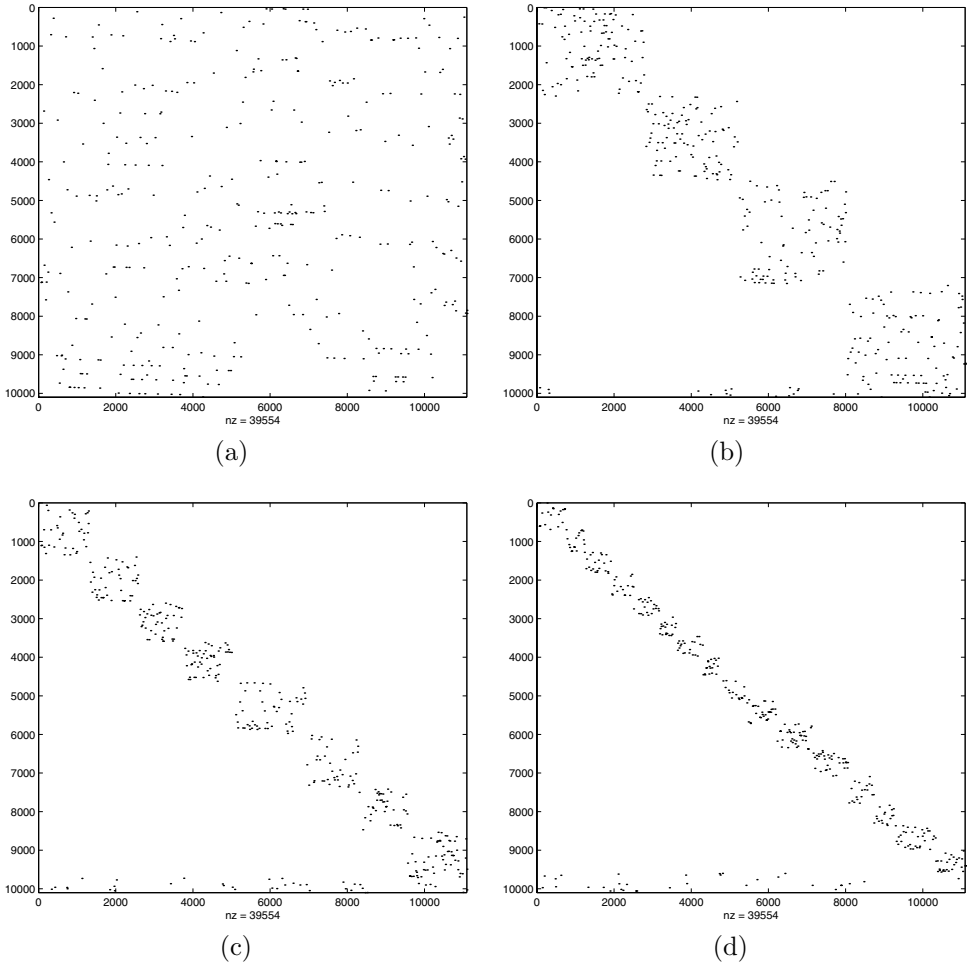


FIG. 7.1. *Rectangular GE matrix with* 10,099 *rows and* 11,098 *columns:* (a) *original structure,* (b) 4*-way SB form,* (c) 8*-way SB form,* (d) 16*-way SB form.*

TABLE 7.2
Performance of different techniques on the bipartite-graph (BG) model.

K	Indirect GPVS			Direct GPVS		
	BG-model (FH)			BG-model (*onmetis*)		
	A_{DB}		A_{SB}	A_{DB}		A_{SB}
	%M$_c$	%N$_c$	%M$_c$	%M$_c$	%N$_c$	%M$_c$
4	6.55	0.20	6.80	1.31	0.22	1.60
8	9.70	0.54	10.40	2.75	0.65	3.60
16	12.79	1.05	14.12	4.15	1.17	5.90
avg	9.68	0.60	10.44	2.74	0.68	3.70

TABLE 7.3
Effect of different balancing criteria in the performance of PaToH.

	R-PaToH			(R+C)-PaToH			(R&C)-PaToH		
K	%M$_c$	%RI	%CI	%M$_c$	%RI	%CI	%M$_c$	%RI	%CI
4	1.62	9.1	15.0	1.69	10.1	10.2	1.72	8.2	10.1
8	3.15	15.6	26.3	3.31	16.7	16.6	3.43	14.5	17.2
16	4.79	23.5	37.1	4.98	25.6	23.9	5.17	21.3	24.6
avg	3.19	16.1	26.2	3.33	17.4	16.9	3.44	14.7	17.3

Table 7.2 presents the results of our experiments on the bipartite graph (BG) 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 in the FH method and direct GPVS tool *onmetis*. Note that FH corresponds to our implementation of the algorithm proposed by Ferris and Horn [12], where we used *kmetis* to partition the bipartite 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. Both schemes produce DB forms with comparable row and column imbalance values. As seen in Table 7.2, the direct *onmetis* scheme produces substantially better DB forms than the indirect FH scheme. Table 7.2 also displays the effect of the column-splitting process used in the second phase of two-phase approaches. In the table, $(\%M_c^{SB} - \%M_c^{DB})/\%N_c = (M_c^{SB} - M_c^{DB})/N_c$ shows the average number of coupling rows induced by a coupling column during the A_{DB}-to-A_{SB} transformation. It can easily be derived from the table that a coupling column induces 1.27 and 1.41 coupling rows in the FH and BG-onmetis schemes, respectively, on average. This means that vertex separators found by these two schemes contain column vertices with small degree, e.g., 2.27 and 2.41. It is interesting to note that both schemes produce DB forms with wide row borders and narrow column borders in general.

For this work, we enhanced PaToH for maintaining different balance criteria that might be used in balancing diagonal blocks of the SB forms. Table 7.3 illustrates the effect of these different balancing criteria in the performance of PaToH. R-PaToH maintains balance on the number of internal nets of the parts during partitioning. (R+C)-PaToH maintains balance on the sum of internal net and vertex counts of the parts during partitioning. (R&C)-PaToH maintains balance on both the number of internal nets and vertices 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 7.3, 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

TABLE 7.4
Performance comparison of the hypergraph model (H-model) with the bipartite graph model (BG-model) in A-to-A_{SB} transformation in terms of the border size (%M_c).

Name	K	H-model	BG-model	
		PaToH	*onmetis*	FH
NL	4	5.02	5.22	27.71
	8	6.02	6.59	32.57
	16	7.19	8.31	36.72
CQ9	4	2.87	2.92	23.06
	8	4.10	4.03	27.76
	16	5.40	5.28	30.50
GE	4	3.01	2.53	4.71
	8	4.37	4.39	8.06
	16	5.63	5.97	10.81
CO9	4	2.72	2.78	21.27
	8	3.78	3.85	26.12
	16	5.10	5.03	30.26
car4	4	0.00	0.00	0.00
	8	0.00	0.52	1.29
	16	0.00	1.83	1.29
fxm4-6	4	0.64	0.41	0.49
	8	1.17	0.80	1.70
	16	2.13	1.42	2.28
fome12	4	0.00	0.00	0.00
	8	9.43	12.27	17.04
	16	15.39	21.23	29.02
pltexpA4-6	4	1.62	0.79	1.08
	8	3.02	2.15	1.98
	16	5.32	4.42	4.77
kent	4	0.34	0.15	0.66
	8	0.70	0.56	2.11
	16	1.26	1.33	3.47
world	4	1.08	0.80	1.53
	8	2.25	2.25	3.79
	16	5.25	5.94	9.29
mod2	4	0.86	0.78	0.88
	8	2.12	2.05	3.42
	16	5.10	5.64	8.75
lpl1	4	3.27	4.08	6.37
	8	5.40	6.58	9.03
	16	6.17	8.76	15.96
fxm3-16	4	0.52	0.33	0.56
	8	0.66	0.73	0.34
	16	0.86	1.51	0.39
Averages over K				
	4	1.69	1.60	6.80
	8	3.31	3.60	10.40
	16	4.98	5.90	14.12
	all	3.33	3.70	10.44

reduced solution space with increasing complexity of the balancing criterion.

Tables 7.4–7.6 present performance comparison of different schemes on A-to-A_{SB} transformation. Tables 7.4 and 7.5 display the quality of SB forms in terms of border size (%M_c) and diagonal-block imbalance (%RI and %CI), respectively, whereas Table 7.6 displays the runtime performance. The FH algorithm effectively maintains 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

TABLE 7.5
Performance comparison of the hypergraph model (H-model) with the bipartite graph model (BG-model) in A-to-A_{SB} transformation in terms of the diagonal-block imbalance size.

Name	K	H-model		BG-model			
		(R+C)-PaToH		*onmetis*		FH	
		%RI	%CI	%RI	%CI	%RI	%CI
NL	4	8.6	6.6	11.8	12.2	15.5	14.0
	8	13.0	11.3	17.6	18.5	23.4	19.2
	16	18.3	15.3	23.7	24.5	28.9	22.2
CQ9	4	17.0	22.6	17.8	17.6	19.5	17.8
	8	26.6	31.0	24.4	25.3	22.9	24.0
	16	37.3	38.6	36.7	29.5	29.5	24.8
GE	4	14.8	11.8	15.5	15.4	13.5	12.1
	8	21.5	19.8	19.3	20.0	19.0	19.4
	16	29.9	27.6	27.0	27.7	28.2	22.9
CO9	4	10.9	19.3	14.7	12.1	18.8	17.1
	8	14.4	27.5	21.2	16.9	20.7	24.4
	16	27.9	33.0	30.1	22.2	26.7	25.4
car4	4	0.6	0.9	3.3	5.9	22.6	25.0
	8	0.6	2.0	12.8	18.7	0.9	2.9
	16	0.7	4.3	23.7	36.1	0.9	6.1
fxm4-6	4	10.0	9.5	2.6	2.4	8.0	7.8
	8	14.7	13.8	10.9	11.0	14.8	14.3
	16	23.2	22.5	19.1	20.2	15.1	15.1
fome12	4	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
	16	19.4	13.8	24.9	22.3	25.9	16.8
pltexpA4-6	4	5.9	4.8	2.9	3.2	11.4	11.7
	8	12.9	10.4	10.2	10.9	11.9	12.5
	16	19.7	17.0	16.2	18.2	15.5	16.7
kent	4	12.2	16.1	12.9	23.8	18.3	21.8
	8	19.3	24.6	21.3	35.0	22.9	18.8
	16	26.7	41.7	31.8	48.5	28.8	32.9
world	4	9.8	11.5	10.3	10.0	10.5	10.0
	8	17.8	20.6	17.8	19.8	15.4	17.5
	16	30.9	28.3	31.0	30.4	22.6	20.0
mod2	4	9.6	10.3	10.6	10.6	11.8	10.7
	8	17.2	18.3	18.4	20.4	15.0	17.1
	16	30.2	26.7	28.7	29.6	22.0	20.4
lpl1	4	18.4	6.8	11.7	5.5	13.4	12.3
	8	31.3	12.0	15.8	11.7	24.1	17.3
	16	40.5	16.0	26.0	18.9	35.8	20.3
fxm3-16	4	13.6	12.9	0.6	0.5	7.8	7.6
	8	17.6	16.6	1.3	1.2	2.4	1.8
	16	27.7	26.4	2.9	2.6	4.6	3.7
Averages over K							
	4	10.1	10.2	8.8	9.2	13.2	12.9
	8	16.7	16.6	15.7	16.9	16.2	15.6
	16	25.6	23.9	24.8	25.4	21.9	19.0
	all	17.4	16.9	16.4	17.2	17.1	15.8

a common experimental framework, the results of the (R+C)-PaToH, BG-onmetis, and FH schemes are displayed in Tables 7.4–7.6.

As seen in Table 7.4, 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 PaToH, on overall average. The one-phase approach PaToH produces approximately 11% fewer coupling rows than the two-phase approach BG-onmetis, on average, which confirms the effectiveness of

TABLE 7.6

Execution times of the partitioning algorithms given in Table 7.5 as percents of the solution times of the LP problems by LOQO. Values in parentheses are the LP solution times in seconds.

Name	*LOQO* sol. time	K	H-model	BG-model	
			PaToH	*onmetis*	FH
NL	100 (804)	4	0.211	0.140	0.090
		8	0.244	0.179	0.090
		16	0.271	0.199	0.106
CQ9	100 (554)	4	0.459	0.339	0.213
		8	0.571	0.447	0.229
		16	0.672	0.538	0.263
GE	100 (403)	4	0.220	0.273	0.136
		8	0.294	0.387	0.154
		16	0.392	0.449	0.169
CO9	100 (708)	4	0.390	0.305	0.189
		8	0.484	0.393	0.205
		16	0.545	0.472	0.233
car4	100 (56)	4	3.562	45.958	46.603
		8	5.168	50.529	54.329
		16	6.704	52.429	58.326
fxm4-6	100 (191)	4	1.978	1.976	0.944
		8	2.941	2.931	0.975
		16	3.884	3.817	0.986
fome12	100 (62677)	4	0.015	0.007	0.004
		8	0.024	0.014	0.005
		16	0.028	0.018	0.007
pltexpA4-6	100 (278)	4	1.576	1.470	0.782
		8	2.328	2.277	0.785
		16	3.029	2.810	0.811
kent	100 (618)	4	0.756	0.898	0.451
		8	1.117	1.333	0.487
		16	1.385	1.662	0.534
world	100 (1163)	4	0.427	0.317	0.169
		8	0.612	0.478	0.178
		16	0.786	0.667	0.214
mod2	100 (1076)	4	0.453	0.334	0.178
		8	0.632	0.509	0.186
		16	0.843	0.710	0.221
lpl1	100 (3800)	4	0.833	0.341	0.169
		8	1.086	0.482	0.178
		16	1.221	0.662	0.198
fxm3-16	100 (449)	4	1.365	1.387	0.719
		8	2.087	2.026	0.690
		16	2.737	2.652	0.659
Averages over K					
		4	0.942	4.134	3.896
		8	1.353	4.768	4.499
		16	1.730	5.160	4.825
		all	1.342	4.688	4.407

the hypergraph model to permute rectangular matrices into SB forms. As seen in Table 7.4, the numbers of coupling rows of the SB forms produced by PaToH remain below 5% for 16-way partitionings, on average. As seen in Tables 7.4–7.5, our methods find balanced permutations, with very few coupling rows, which would lead to efficient parallel solutions.

Table 7.6 displays execution times of the partitioning algorithms as percents of the solution times of the respective LP problems by LOQO [43]. As seen in this table, partitioning times are affordable when compared with the LP solution times. For

example, LOQO [43] solves the *lpl*1 problem, which has the constraint matrix with the largest $M \times N$ product, in approximately 3800 seconds. As seen in Table 7.6, the 16-way partitioning times of all algorithms remain below 1.22% of the LOQO solution time of this LP problem. As also seen in the table, partitioning times of all algorithms remain well below 4% of the LOQO solution times of all LP problems except *car*4.

In two-phase approaches, hypergraph and bipartite representations of a rectangular matrix are of equal size: the number of nonzeros in the matrix. However, the clustering phase of an HP tool involves more costly operations than those of a GP tool. Hence, two-phase approaches using a GP tool are expected to run faster than the one-phase approach using an HP tool. As seen in Table 7.6, the two-phase approach BG-onmetis runs faster than PaToH in the partitioning of all test matrices except GE, *car*4, and *kent*.

8. Conclusion. We investigated 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. 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 overview solution techniques and tools for solving the stated problems. Experimental results on a wide range of sparse matrices were impressive and showed that our methods can effectively extract the underlying block-diagonal structure of a matrix.

REFERENCES

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

[2] C. Ashcraft and J. W. H. Liu, *Applications of the Dulmage–Mendelsohn decomposition and network flow to graph bisection improvement*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 325–354.

[3] C. Aykanat, A. Pinar, and U. V. Çatalyürek, *Permuting Sparse Rectangular Matrices into Block-Diagonal Form*, tech. report, Department of Computer Engineering, Bilkent University, Ankara, Turkey, 2002.

[4] Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.

[5] T. N. Bui and C. Jones, *Finding good approximate vertex and edge partitions is NP-hard*, Inform. Process. Lett., 42 (1992), pp. 153–159.

[6] T. N. Bui and C. Jones, *A heuristic for reducing fill-in in sparse matrix factorization*, in Proceedings of the 6th SIAM Conference on Parallel Processing for Scientific Computing, SIAM, Philadelphia, 1993, pp. 445–452.

[7] U. V. Çatalyürek and C. Aykanat, *Decomposing irregularly sparse matrices for parallel matrix-vector multiplications*, in Proceedings of the 3rd International Symposium on Solving Irregularly Structured Problems in Parallel, Irregular'96, Lecture Notes in Comput. Sci. 1117, Springer-Verlag, Berlin, 1996, pp. 75–86.

[8] U. V. Çatalyürek and C. Aykanat, *Hypergraph-partitioning based decomposition for parallel sparse-matrix vector multiplication*, IEEE Trans. Parallel Distrib. Systems, 10 (1999), pp. 673–693.

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

[10] I. O. Center, *Linear Programming Problems*, ftp://col.biz.uiowa.edu:pub/testprob/lp/gondzio.

[11] G. Dantzig and P. Wolfe, *Decomposition principle for linear programs*, Oper. Res., 8 (1960), pp. 101–111.
[12] M. C. Ferris and J. D. Horn, *Partitioning mathematical programs for parallel solution*, Math. Programming, 80 (1998), pp. 35–61.
[13] C. M. Fiduccia and R. M. Mattheyses, *A linear-time heuristic for improving network partitions*, in Proceedings of the 19th ACM/IEEE Design Automation Conference, IEEE Press, Piscataway, NJ, 1982, pp. 175–181.
[14] A. George, *Nested dissection of a regular finite element mesh*, SIAM J. Numer. Anal., 10 (1973), pp. 345–363.
[15] S. K. Gnanendran and J. K. Ho, *Load balancing in the parallel optimization of block-angular linear programs*, Math. Programming, 62 (1993), pp. 41–67.
[16] A. Gupta, *Watson Graph Partitioning Package*, Tech. Report RC 20453, IBM T. J. Watson Research Center, Yorktown Heights, NY, 1996.
[17] S. Hauck and G. Boriello, *An evaluation of bipartitioning techniques*, IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, 16 (1997), pp. 849–866.
[18] B. Hendrickson, *Graph partitioning and parallel solvers: Has the emperor no clothes?*, in Proceedings of the 5th International Symposium on Solving Irregularly Structured Problems in Parallel, Irregular'98, Lecture Notes in Comput. Sci. 1457, Springer-Verlag, Berlin, 1998, pp. 218–225.
[19] B. Hendrickson and T. G. Kolda, *Graph partitioning models for parallel computing*, Parallel Comput., 26 (2000), pp. 1519–1534.
[20] B. Hendrickson and R. Leland, *A Multilevel Algorithm for Partitioning Graphs*, tech. report, Sandia National Laboratories, Albuquerque, NM, 1993.
[21] B. Hendrickson and R. Leland, *The Chaco User's Guide, Version* 2.0, Sandia National Laboratories, Albuquerque, NM, 1995.
[22] B. Hendrickson and E. Rothberg, *Improving the run time and quality of nested dissection ordering*, SIAM J. Sci. Comput., 20 (1998), pp. 468–489.
[23] J. K. Ho, T. C. Lee, and R. P. Sundarraj, *Decomposition of linear programs using parallel computation*, Math. Programming, 42 (1988), pp. 391–405.
[24] Y. F. Hu, C. M. Maguire, and R. J. Blake, *Ordering unsymmetric matrices into bordered block diagonal form for parallel processing*, in Proceedings of Euro-Par '99 Parallel Processing: 5th International Euro-Par Conference, Toulouse, France, 1999, P. Amestoy, P. Berger, M. J. Daydé, I. S. Duff, V. Frayssé, L. Giraud, and D. Ruiz, eds., Lecture Notes in Comput. Sci. 1685, Springer-Verlag, Berlin, 1999, pp. 295–302.
[25] *Hungarian Academy of Sciences: Computer and Automation Research Institute, LP Test Sets*, ftp://ftp.sztaki.hu/pub/oplab/LPTESTSET/.
[26] *IMSL User's Manual, Edition* 9.2 *(International Mathematical and Statistical Library)*, Houston, TX, 1984.
[27] G. Karypis and V. Kumar, *MeTiS. A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices Version* 3.0, Department of Computer Science and Engineering/Army HPC Research Center, University of Minnesota, Minneapolis, MN, 1998.
[28] G. Karypis and V. Kumar, *Multilevel Algorithms for Multi-constraint Graph Partitioning*, Tech. Report 98-019, Department of Computer Science/Army HPC Research Center, University of Minnesota, Minneapolis, MN, 1998.
[29] G. Karypis, V. Kumar, R. Aggarwal, and S. Shekhar, *hMeTiS. A Hypergraph Partitioning Package Version* 1.0.1, Department of Computer Science and Engineering/Army HPC Research Center, University of Minnesota, Minneapolis, MN, 1998.
[30] B. W. Kernighan and S. Lin, *An efficient heuristic procedure for partitioning graphs*, Bell System Tech. J., 49 (1970), pp. 291–307.
[31] C. Lemarechal, A. Nemirovski, and Y. Nesterov, *New variants of bundle methods*, Math. Programming, 69 (1995), pp. 111–147.
[32] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*, Wiley–Teubner, Chichester, UK, 199.
[33] J. W. H. Liu, *The minimum degree ordering with constraints*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 1136–1145.
[34] D. Medhi, *Parallel bundle-based decomposition for large-scale structured mathematical programming problems*, Ann. Oper. Res., 22 (1990), pp. 101–127.
[35] D. Medhi, *Bundle-based decomposition for structured large-scale convex optimization: Error estimate and application to block-angular linear programs*, Math. Programming, 66 (1994), pp. 79–101.
[36] R. R. Meyer and G. Zakeri, *Multicoordination methods for solving convex block-angular*

programs, SIAM J. Optim., 10 (1999), pp. 121–131.

[37] J. M. Mulvey and A. Ruszcynski, *A diagonal quadratic approximation method for large scale linear programs*, Oper. Res. Lett., 12 (1992), pp. 205–215.

[38] A. Pinar, U. V. Çatalyürek, C. Aykanat, and M. Pinar, *Decomposing linear programs for parallel solution*, in Proceedings of the Second International Workshop on Applied Parallel Computing, PARA '95, Lyngby, Denmark, 1995, Lecture Notes in Comput. Sci. 1041, Springer-Verlag, Berlin, 1996, pp. 473–482.

[39] A. Pinar and C. Aykanat, *An effective model to decompose linear programs for parallel solution*, in Proceedings of the Third International Workshop on Applied Parallel Computing, PARA'96, Lecture Notes in Comput. Sci. 1184, Springer-Verlag, Berlin, 1997, pp. 592–601.

[40] A. Pinar and B. Hendrickson, *Partitioning for complex objectives*, in Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS), San Francisco, CA, IEEE Computer Society Press, Los Alamitos, CA, 2001.

[41] T. Rashid and T. A. Davis, *An approach for parallelizing any general unsymmetric sparse matrix algorithm*, in Proceedings of the 7th SIAM Conference on Parallel Processing for Scientific Computing, SIAM, Philadelphia, 1995, pp. 413–417.

[42] J. M. Stern and S. A. Vavasis, *Active set algorithms for problems in block angular form*, Comput. Appl. Math., 12 (1994), pp. 199–226.

[43] R. J. Vanderbei, *LOQO User's Manual, Version* 4.01, Tech. Report SOR 97-08, Princeton University, Princeton, NJ, 1997.