Technical Report

TR-2006-010

Revisiting hypergraph models for sparse matrix decomposition

by

Cevdet Aykanat, Bora Ucar

Mathematics and Computer Science

EMORY UNIVERSITY

REVISITING HYPERGRAPH MODELS FOR SPARSE MATRIX DECOMPOSITION*

BORA UÇAR† AND CEVDET AYKANAT‡

Abstract. We provide an exposition of the hypergraph models for parallel sparse matrix-vector multiplies based on one-dimensional (1D) matrix partitioning. Our aim is to emphasize the expressive power of the hypergraph models. We first set forth an elementary hypergraph model in which vertices represent the data elements of a matrix-vector multiply operation and nets encode data dependencies. We then apply a recently proposed hypergraph transformation operation to devise models for 1D sparse matrix decomposition. The resulting 1D partitioning models are equivalent to the previously proposed computational hypergraph models and are not meant to be replacements for them. Nevertheless, the new models give us insights into the previous ones and help us explain a subtle requirement, known as the consistency condition, of the hypergraph partitioning models. We also demonstrate the flexibility of the elementary model on a few partitioning problems that are hard to solve using the previously proposed models.

Key words. parallel computing, sparse matrix-vector multiply, hypergraph models

AMS subject classifications. 05C50, 05C65, 65F10, 65F50, 65Y05

1. Introduction. Computational hypergraph models for one-dimensional (1D) sparse matrix decomposition proposed in [3, 4, 9] have gained widespread acceptance. These models can address partitionings of rectangular, unsymmetric square, and symmetric square matrices. However, the expressive power of these models had been acknowledged long after their introduction [1, 7, 11]. This is due to three reasons. First, the works [3, 9] had limited distribution, and therefore, the models seem to be introduced in [4]. Second, the paper [4] does not discuss rectangular matrices explicitly. Third, perhaps the most probable one, is that the models introduced in [4] address symmetric partitioning and require zero-free diagonals (consistency condition) for establishing an exact correspondence between the total volume of communication and the hypergraph partitioning objective. The consistency condition—the vertex v_i should be in the net n_i—evokes square matrices.

In §3, we present an elementary hypergraph model for 1D sparse matrix decomposition for parallel sparse matrix-vector multiplies. The model represents all the operands of the sparse matrix-vector multiply operation $y \leftarrow Ax$ with vertices. Therefore, partitioning the proposed hypergraph model amounts to partitioning the input vector x, the output vector y, and the rows or columns of A. We show that this elementary model can be transformed into hypergraph models for obtaining unsymmetric and symmetric partitionings. The resulting models are equivalent to the previously proposed computational hypergraphs in modeling the total volume of communication correctly. If symmetric partitioning is sought, the resulting model becomes topologically identical to the previously proposed models [4]. However, there

*This work was partially supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK) under grant 103E028.

†Department of Mathematics and Computer Science, Emory University, Atlanta, Georgia 30322, USA (ubora@mathcs.emory.edu). The work of this author is partially supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK) under the program 2219 and by the University Research Committee of Emory University.

‡Bilkent University Computer Engineering Department, Ankara, 06800, Turkey (aykanat@cs.bilkent.edu.tr)

is a slight discrepancy between the significance of the nets in these two topologically identical models. This discrepancy helps us automatically satisfy the consistency condition even for matrices with missing diagonal entries without referring to or altering the sparsity pattern of the matrix A.

Although the elementary model contributes a little in the standard 1D matrix partitioning, it is useful in general. In §4, we show how to transform the elementary model to address a few partitioning problems that are hard to tackle using the previous models. We confine the discussion to the rowwise partitioning models, because the columnwise partitioning models can be addressed similarly.

2. Background. A hypergraph $\mathcal{H} = (\mathcal{V},\mathcal{N})$ is defined as a set of vertices $\mathcal{V}$ and a set of nets $\mathcal{N}$. Every net is a subset of vertices. The vertices of a net are also called its *pins*. The size of a net n_i is equal to the number of its pins, i.e., $|n_i|$. The set of nets that contain vertex v_j is denoted by $Nets(v_j)$. Weights can be associated with vertices. We use $w(j)$ to denote the weight of the vertex v_j.

$\Pi = \{\mathcal{V}_1, \ldots, \mathcal{V}_K\}$ is a K-way vertex partition of $\mathcal{H} = (\mathcal{V},\mathcal{N})$ if each part is nonempty, parts are pairwise disjoint, and the union of parts gives $\mathcal{V}$. In Π, a net is said to *connect* a part if it has at least one pin in that part. The *connectivity set* $\Lambda(i)$ of a net n_i is the set of parts connected by n_i. The *connectivity* $\lambda(i) = |\Lambda(i)|$ of a net n_i is the number of parts connected by n_i. A net is said to be cut if it connects more than one part and uncut otherwise. In Π, the weight of a part is the sum of the weights of vertices in that part.

In the hypergraph partitioning problem, the objective is to minimize

$$cutsize(\Pi) = \sum_{n_i \in \mathcal{N}} (\lambda(i) - 1) \; . \tag{2.1}$$

This objective function is widely used in VLSI community [8] and in scientific computing community [1, 4, 11], and it is referred to as the *connectivity* -1 cutsize metric. The partitioning constraint is to satisfy a balancing constraint on part weights:

$$\frac{W_{max} - W_{avg}}{W_{avg}} \leq \epsilon \; . \tag{2.2}$$

Here W_{max} is the weight of the part with the maximum weight, W_{avg} is the average part weight, and ϵ is a predetermined imbalance ratio. This problem is NP-hard [8].

We make use of the recently proposed vertex amalgamation operation [12]. This operation combines two vertices into a single composite vertex. The nets of the resulting composite vertex are set to the union of the nets of the constituent vertices, e.g., amalgamating vertices v_i and v_j removes these two vertices from the hypergraph, adds a new vertex v_{ij}, and sets $Nets(v_{ij}) = Nets(v_i) \cup Nets(v_j)$.

3. Revisiting hypergraph models for 1D partitioning. Consider the computations of the form $y \leftarrow Ax$ under rowwise partitioning of the $m \times n$ matrix A. Since we partition the rows of A, the entries of the input- and output-vectors x and y, there should be three types of vertices in a hypergraph: row-vertices, x-vertices, and y-vertices. The nets of the hypergraph should be defined to represent the dependencies of the y-vertices on the row-vertices, and the dependencies of the row-vertices on the x-vertices. We define the hypergraph $\mathcal{H} = (\mathcal{V},\mathcal{N})$ with $|\mathcal{V}| = 2m + n$ vertices and $|\mathcal{N}| = m + n$ nets. The vertex set $\mathcal{V} = \mathcal{X} \cup \mathcal{Y} \cup \mathcal{R}$ contains the vertices $\mathcal{X} = \{x_1, \ldots, x_n\}$, $\mathcal{Y} = \{y_1, \ldots, y_m\}$, and $\mathcal{R} = \{r_1, \ldots, r_m\}$. Here x_j corresponds to the jth entry in the input-vector, y_i corresponds to the ith entry in the output-vector,

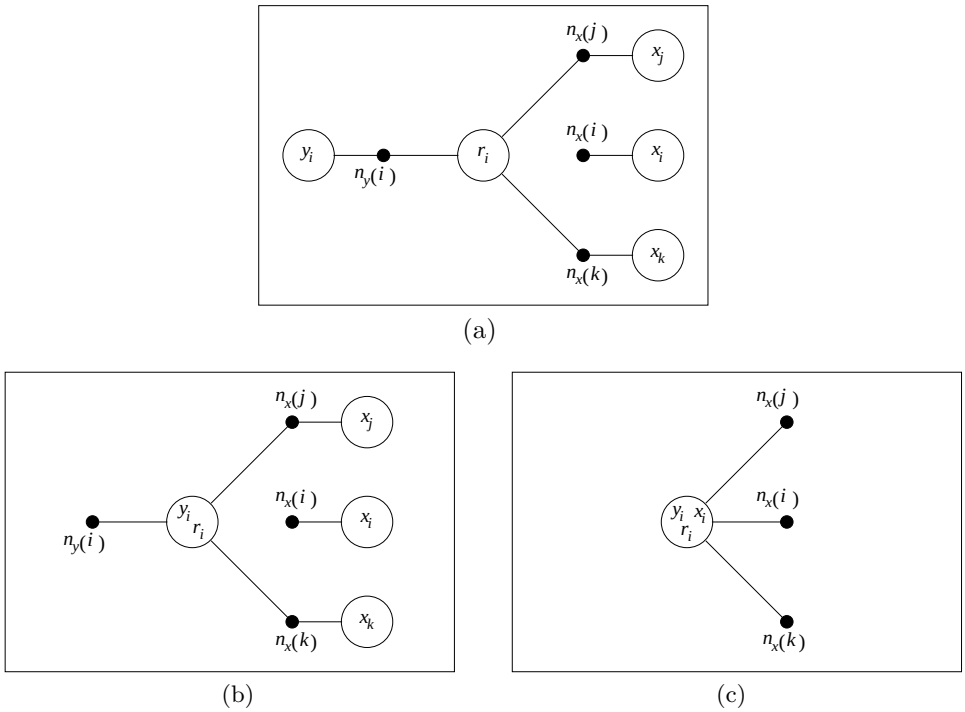

FIG. 3.1. *(a) A general 1D partitioning model in which all operands of the matrix-vector multiply operation are represented by vertices. (b) 1D unsymmetric partitioning model is obtained by applying the vertex amalgamation operation on y_i and r_i to enforce the owner-computes rule. (c) 1D symmetric partitioning model is obtained by applying the vertex amalgamation operation to the composite vertex y_i/r_i and the vertex x_i.*

and r_i corresponds to the ith row of A. The net set $\mathcal{N} = \mathcal{N}_x \cup \mathcal{N}_y$ contains the nets $\mathcal{N}_x = \{n_x(j) : j = 1, \ldots, n\}$ where $n_x(j) = \{r_i : i = 1, \ldots, m \text{ and } a_{ij} \neq 0\} \cup \{x_j\}$ and the nets $\mathcal{N}_y = \{n_y(i) : i = 1, \ldots, m\}$ where $n_y(i) = \{y_i, r_i\}$. In accordance with the previous models [4], each row-vertex r_i is associated with a weight to represent the computational load associated with the ith row, e.g., $w_r(i) = |Nets(r_i)| - 1$. Note that the weight $w_r(i)$ corresponds to the number of nonzeros in the ith row of A as in [4]. Weights can be associated with the x- and y-vertices. For example, a unit weight may be assigned to these vertices in order to maintain balance in linear vector operations. Figure 3.1(a) shows a portion of the elementary hypergraph built accordingly. In the figure, the ith row has two nonzeros: one in the jth column and the other in the kth column. Hence the row-vertex r_i is connected to the nets $n_y(i)$, $n_x(j)$, and $n_x(k)$. The nets $n_x(i)$, $n_x(j)$, and $n_x(k)$ contain the respective x-vertices and some row-vertices which are not shown for the sake of clarity.

Observe that in the above construction, each net contains a unique vertex that is either a source or a target, e.g., y_i or x_i. This construction abides by the guidelines given in [4] and outlined in [7]. The elementary hypergraph model is the most general model for 1D rowwise partitioning, since by partitioning the vertices of this hypergraph we can obtain partitions on all operands of the matrix-vector multiply operation.

Now, we show how to modify the elementary hypergraph by applying the vertex amalgamation operation to devise 1D unsymmetric and symmetric partitionings.

First, we can apply the owner-computes rule, i.e., y_i should be computed by the processor that owns r_i. This requires amalgamating the vertices y_i and r_i for all i. A portion of the resulting hypergraph is shown in Fig. 3.1(b). Since nets of size one do not contribute to the partitioning cost, we can delete the net $n_y(i)$ from the model. Partitioning the resulting hypergraph will produce nonsymmetric partitions. Suppose we are seeking symmetric partitions, i.e., the processor which owns r_i and y_i should own x_i. This time, we have to amalgamate the vertices y_i/r_i and x_i for all i. A portion of the resulting hypergraph is shown in Fig. 3.1(c). Partitioning the resulting hypergraph will produce symmetric partitions. Note that the hypergraph shown in Fig. 3.1(c) is topologically identical to the column-net hypergraph model proposed in [4]. However, there is a difference in the semantics. Here, the x-vector entries are represented by the vertices, whereas in [4] they are represented by the nets. Since the vertex amalgamation operation between the vertex y_i/r_i and x_i connects the ith vertex to the ith net, this difference guarantees the consistency condition without referring to or altering the sparsity pattern of the matrix.

4. Examples. We cast three partitioning problems which are hard to solve using the previous models. Each problem asks for a distinct hypergraph model whose cutsize under a partition corresponds to the total volume of communication in parallel computations with a proper algorithm.

Problem 1. *Describe a hypergraph model which can be used to partition the matrix A rowwise for the $y \leftarrow Ax$ computations under given, possibly different, partitions on the input- and output-vectors x and y.*

A parallel algorithm that carries out the $y \leftarrow Ax$ computations under given partitions of x and y should have a communication phase on x, a computation phase, and a communication phase on y. We take the elementary hypergraph model given in Fig. 3.1(a) and then designate each x_j and y_i as fixed to a part according to the given partitions on the vectors x and y. Invoking a hypergraph partitioning tool which can handle the fix vertices (e.g., PaToH [5]) will solve the partitioning problem stated above. For each $n_x(j)$, the *connectivity* -1 value, i.e., $\lambda_x(j)-1$, corresponds to the total volume of communication regarding x_j. Similarly, for each $n_y(i)$, $\lambda_y(i)-1$ corresponds to the volume of communication regarding y_i; note $\lambda_y(i)-1$ is either 0 (r_i is assigned to the part to which y_i is fixed) or 1 (otherwise).

Problem 2. *Describe a hypergraph model to obtain the same partition on the input- and output-vectors x and y which is different than the partition on the rows of A for the $y \leftarrow Ax$ computations.*

The computations $y \leftarrow Ax$ should be carried out by the algorithm given for Problem 1. We take the elementary hypergraph model given in Fig. 3.1(a) and then amalgamate the vertices y_i and x_i into a single vertex. The portion of the resulting hypergraph is shown in Fig. 4.1. Here, the *connectivity* -1 values of the nets again correspond to the volume of communication regarding the associated x- and y-vector entries. The communications on x_i are still represented by the net $n_x(i)$, and the communications on y_i are still represented by the net $n_y(i)$. Observe that a composite vertex y_i/x_i can be in the same part with r_i in which case there is no communication on y_i and $\lambda_y(i)-1=0$.

Problem 3. *Describe a hypergraph model to obtain different partitions on x and on the rows of A, where y is partitioned conformably with the rows of A under the owner-computes rule for computations of the form $y \leftarrow Ax$ followed by $x \leftarrow x+y$.*

The computations $x_i + y_i$ introduce new vertices for all i. The vertex $x_i + y_i$ depends on the vertices x_i and y_i. Therefore, it is connected to the nets $n_x(i)$ and

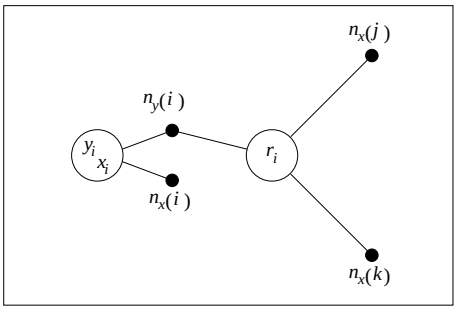

FIG. 4.1. *Hypergraph model for the partitioning problem 2.*

$n_y(i)$. Furthermore, since x_i is dependent on the vertex $x_i + y_i$, we connect x_i to the net $n_{x+y}(i)$. A portion of the hypergraph that encapsulates these dependencies is shown in Fig. 4.2(a). First, we enforce the owner-computes rule for the computations $x_i \leftarrow x_i + y_i$. This can be achieved by amalgamating the vertices x_i and $x_i + y_i$. Since, the size of the net $n_{x+y}(i)$ becomes one, it can be excluded safely. The resulting model is shown in Fig. 4.2(b). Next, we enforce the owner-computes rule for y_i by amalgamating vertices y_i and r_i (Fig. 4.2(c)). In order to carry out the computations $x_i \leftarrow x_i + y_i$, the y_i values should be communicated after computing $y \leftarrow Ax$. Here, if the composite vertex $x_i/x_i + y_i$ and the composite vertex r_i/y_i reside in different processors, then we have to send y_i. The communication volume of this send operation is equal to $\lambda_y(i) - 1 = 1$. Since the nets in $\mathcal{N}_x$ are kept intact, they represent the communications on the x-vector entries for the $y \leftarrow Ax$ computations as before.

Consider a slightly different partitioning problem in which we do not need the owner-computes rule for the y-vector entries. The hypergraph in Fig. 4.2(b) can be used to address this partitioning problem. Here, if x_i, y_i, and r_i reside in different processors, then we will have two units of communication: the result of the inner product $\langle r_i, x \rangle$ will be sent to the processor that holds y_i which will write y_i and send the value to the processor that holds x_i. If, however, x_i and r_i reside in the same processor, we will have one unit of communication: the result $\langle r_i, x \rangle$ will be sent to the processor that holds y_i and the computation $x_i \leftarrow x_i + y_i$ will be performed using the local data x_i and $\langle r_i, x \rangle$.

5. Discussion. We provided an elementary hypergraph model to partition the data of the $y \leftarrow Ax$ computations. The model represents all operands of the matrix-vector multiply operation as vertices. Therefore, partitioning the vertices of this elementary model amounts to partitioning all operands of the multiply operation. We showed how to transform the elementary model into hypergraph models that can be used to address various 1D partitioning problems including the symmetric (same partition on x and y) and unsymmetric (different partitions on x and y) partitioning problems. Although the latter two problems are well studied, the models discussed here shed light into the previous models.

We confined the discussion to rowwise partitioning problems for brevity. The columnwise partitioning models can be constructed similarly. For example, the elementary model for the $y \leftarrow Ax$ computations under columnwise partitioning of A is given by $H_C = (\mathcal{V}, \mathcal{N})$, where $\mathcal{V} = \mathcal{X} \cup \mathcal{Y} \cup \mathcal{C}$ with $\mathcal{X} = \{x_1, \ldots, x_n\}$ corresponding to the input vector entries, $\mathcal{Y} = \{y_1, \ldots, y_m\}$ corresponding to the output vector entries, $\mathcal{C} = \{c_1, \ldots, c_n\}$ corresponding to the columns of A; $\mathcal{N} = \mathcal{N}_x \cup \mathcal{N}_y$

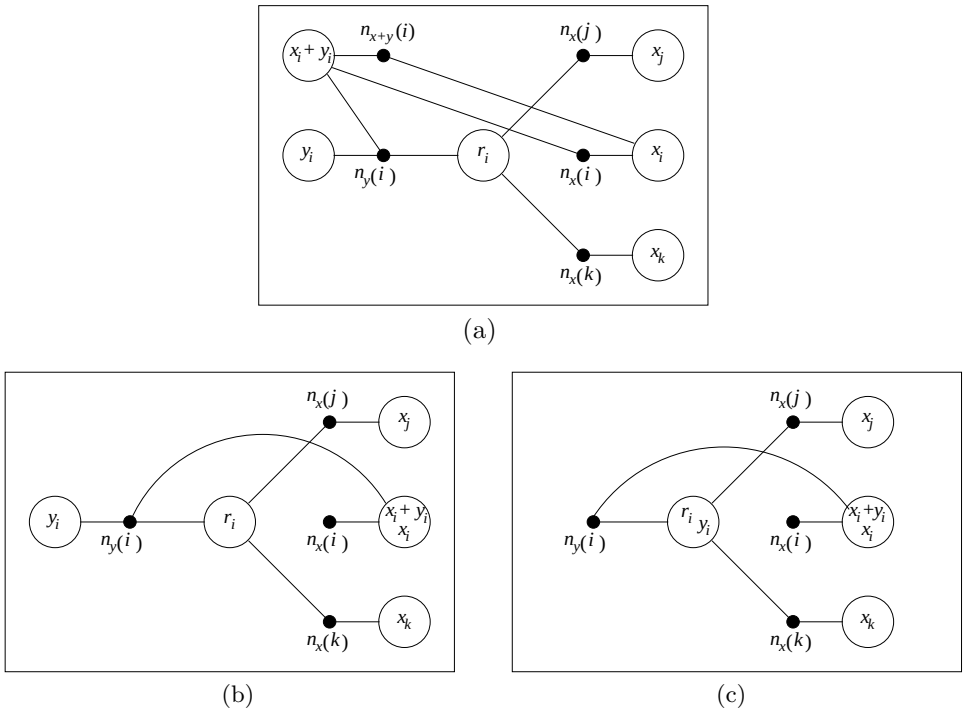

FIG. 4.2. *(a) Basic hypergraph model for the partitioning problem 3. (b) According to the owner-computes rule for the computations $x_i \leftarrow x_i + y_i$, the vertices x_i and $x_i + y_i$ are amalgamated. (c) According to the owner-computes rule for y_i, the vertices y_i and r_i are amalgamated.*

with the nets $\mathcal{N}_x = \{n_x(j) : j = 1, \ldots, n\}$ where $n_x(j) = \{x_j, c_j\}$, and the nets $\mathcal{N}_y = \{n_y(i) : i = 1, \ldots, m\}$ where $n_y(i) = \{c_j : j = 1, \ldots, n \text{ and } a_{ij} \neq 0\} \cup \{y_i\}$. Models for the other problems should follow easily by applying vertex amalgamation operation.

The basic ideas can be carried over to two-dimensional, nonzero-based partitioning model [6] as well. The elementary model for the $y \leftarrow Ax$ computations under fine-grain partitioning of A is given by $H_{2D} = (\mathcal{V}, \mathcal{N})$. The vertex set $\mathcal{V} = \mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z}$ contains the vertices $\mathcal{X} = \{x_1, \ldots, x_n\}$ corresponding to the input vector entries, $\mathcal{Y} = \{y_1, \ldots, y_m\}$ corresponding to the output vector entries, and $\mathcal{Z} = \{a_{ij} : 1 \leq i \leq m \text{ and } 1 \leq j \leq n \text{ and } a_{ij} \neq 0\}$ corresponding to the nonzeros of A. The net set $\mathcal{N} = \mathcal{N}_x \cup \mathcal{N}_y$ contains the nets $\mathcal{N}_x = \{n_x(j) : j = 1, \ldots, n\}$ where $n_x(j) = \{a_{ij} : 1 \leq i \leq m \text{ and } a_{ij} \neq 0\} \cup \{x_j\}$, and $\mathcal{N}_y = \{n_y(i) : i = 1, \ldots, m\}$ where $n_y(i) = \{a_{ij} : 1 \leq j \leq n \text{ and } a_{ij} \neq 0\} \cup \{y_i\}$. Applying the vertex amalgamation operation to vertices x_i and y_i for $1 \leq i \leq n$ (if the matrix is square) yields a model whose partitioning results in symmetric partitioning on the input and output vectors of the $y \leftarrow Ax$ multiply. Notice the resulting model again satisfies the consistency condition. However, this model is slightly different than the original fine-grain model [6]. In order to guarantee the consistency condition, Çatalyürek and Aykanat [6] add a dummy vertex d_{ii} for each diagonal entry a_{ii} that is originally zero in A. After the vertex amalgamation operation, H_{2D} contains n composite vertices of the form (x_i, y_i). If a_{ii} is zero in A, then the vertex (x_i, y_i) can be said to be equivalent to the dummy vertex d_{ii}. If, however, a_{ii} is nonzero in A, then the vertex (x_i, y_i) can be said to be a copy of

the diagonal vertex a_{ii}. Having observed this discrepancy between the models, we have done experiments with a number of matrices. We did not observe any significant difference between the models in terms of cutsize (total volume).

We should mention that the owner-computes rule should be enforced unless otherwise dictated by the problem. Because, it reduces the number of vertices and possible nets leading to a reduction in model size and in the running time of the partitioning algorithm. More importantly, it avoids a communication phase.

Current approach in the parallelization of a wide range of iterative solvers is to enforce the same partition on the vectors that participate in a linear vector operation. This approach avoids a reordering operation—which is bound to be communication intensive—on the vectors. The models provided in this paper can be used to encapsulate the total volume of communication in the vector ordering operation. Therefore, the models can be used to exploit the flexibility in partitioning disjoint phases of computations.

Although the elementary model and subsequent models obtained from it help partition all the operands of a matrix-vector multiply neatly, it conceals the freedom in assigning vector entries to processors to optimize other cost metrics. For example, vertex x_i in Fig. 3.1(b) can be re-assigned to any processor in $\Lambda_x(i)$ without changing the load decomposition to reduce communication cost (see [2, 10, 11, 13]).

Acknowledgment. We thank Prof. R. Bisseling of Utrecht University for his helpful suggestions.

REFERENCES

[1] C. Aykanat, A. Pinar, and Ü. V. Çatalyürek, *Permuting sparse rectangular matrices into block-diagonal form*, SIAM Journal on Scientific Computing, 25 (2004), pp. 1860–1879.

[2] R. H. Bisseling and W. Meesen, *Communication balancing in parallel sparse matrix-vector multiplication*, Electronic Transactions on Numerical Analysis, 21 (2005), pp. 47–65.

[3] Ü. V. Çatalyürek and C. Aykanat, *Decomposing irregularly sparse matrices for parallel matrix-vector multiplications*, Lecture Notes in Computer Science, 1117 (1996), pp. 75–86.

[4] ———, *Hypergraph-partitioning based decomposition for parallel sparse-matrix vector multiplication*, IEEE Transactions Parallel and Distributed Systems, 10 (1999), pp. 673–693.

[5] ———, *PaToH: A multilevel hypergraph partitioning tool, version 3.0*, Tech. Rep. BU-CE-9915, Computer Engineering Department, Bilkent University, 1999.

[6] ———, *A fine-grain hypergraph model for 2d decomposition of sparse matrices*, in Proceedings of 15th International Parallel and Distributed Processing Symposium (IPDPS), San Francisco, CA, April 2001.

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

[8] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*, Wiley–Teubner, Chichester, U.K., 1990.

[9] A. Pinar, Ü. V. Çatalyürek, C. Aykanat, and M. Pinar, *Decomposing linear programs for parallel solution*, Lecture Notes in Computer Science, 1041 (1996), pp. 473–482.

[10] B. Uçar and C. Aykanat, *Minimizing communication cost in fine-grain partitioning of sparse matrices*, Lecture Notes in Computer Science, 2869 (2003), pp. 926–933.

[11] ———, *Encapsulating multiple communication-cost metrics in partitioning sparse rectangular matrices for parallel matrix-vector multiplies*, SIAM Journal on Scientific Computing, 25 (2004), pp. 1827–1859.

[12] ———, *Partitioning sparse matrices for parallel preconditioned iterative methods*, SIAM Journal on Scientific Computing, (submitted) (2004).

[13] B. Vastenhouw and R. H. Bisseling, *A two-dimensional data distribution method for parallel sparse matrix-vector multiplication*, SIAM Review, 47 (2005), pp. 67–95.