REVISITING HYPERGRAPH MODELS FOR SPARSE MATRIX PARTITIONING*

BORA UÇAR† AND CEVDET AYKANAT‡

Abstract. We provide an exposition of hypergraph models for parallelizing sparse matrix-vector multiplies. Our aim is to emphasize the expressive power of hypergraph models. First, we set forth an elementary hypergraph model for parallel matrix-vector multiply based on one-dimensional (1D) matrix partitioning. In the elementary model, the vertices represent the data of a matrix-vector multiply, and the nets encode dependencies among the data. We then apply a recently proposed hypergraph transformation operation to devise models for 1D sparse matrix partitioning. 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. Later, we demonstrate the flexibility of the elementary model on a few 1D partitioning problems that are hard to solve using the previously proposed models. We also discuss extensions of the proposed elementary model to two-dimensional matrix partitioning.

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

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

1. Introduction. Hypergraph-partitioning-based models for parallel sparse matrix-vector multiply operation [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 has only been acknowledged long after their introduction [1, 7, 11]. This may have three main reasons. First, the works [3, 9] had limited distribution, and therefore, the models seem to be introduced in [4]. Second, rectangular matrices are not discussed explicitly in [4]. Third, perhaps the most probable one, is that the paper [4] focuses on obtaining the same partitions on the input and output vectors of the multiply operation. This partitioning scheme evokes square matrices, as the lengths of the input and output vectors have to be the same.

In order to parallelize the matrix-vector multiply $y \leftarrow Ax$, we have to partition the vectors x and y along with the matrix A among the processors of a parallel computer. There are two alternatives in partitioning the vectors x and y. The first one, symmetric partitioning, is to have the same partition on x and y. The second one, unsymmetric partitioning, is to have different partitions on x and y. Usually, if the matrix is partitioned rowwise, the partition on y conforms to the partition on the rows of A. Similarly, if the matrix is partitioned columnwise, the partition on x conforms to the partition on the columns of A.

In §3, we present an elementary hypergraph model for parallel matrix-vector multiply based on one-dimensional (1D) matrix partitioning. The model represents all

*This work was partially supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK) under grant 106E069.

†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).

the operands of the matrix-vector multiply $y \leftarrow Ax$ with vertices. Therefore, partitioning the proposed hypergraph model amounts to partitioning the input vector x, the output vector y, and the matrix A simultaneously. We show that the proposed 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 structurally equivalent to the previously proposed models [4].

Although the elementary model contributes only 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. In most of the paper, 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 size of a net n_i is equal to the number of its vertices, 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 vertex 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. 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 the VLSI community [8] and in the 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 largest part weight, W_{avg} is the average part weight, and ϵ is a predetermined imbalance ratio. This problem is NP-hard [8].

In the previously proposed hypergraph-partitioning-based methods (e.g., [4, 6, 13]), vertices of a hypergraph are used to represent the matrix data (e.g., rows, columns, or nonzeros). Therefore, partitioning the vertices of a hypergraph into K parts amounts to partitioning a matrix among K processors. Usually, the processor P_k is set to be the owner of the data corresponding to the vertices in $\mathcal{V}_k$.

Since the aforementioned approaches do not represent the vector entries with the vertices, they leave the vector partitioning unsolved. Vector partitioning is either done implicitly using the partitions on the matrix for symmetric partitioning [4, 6], or it is done in an additional stage after partitioning the matrix, for unsymmetric [2, 10, 11, 13] and symmetric [10, 13] partitionings. In these models, there is a condition, known as the consistency condition [4], on the exact correspondence between the total communication volume and the hypergraph partitioning objective.

The consistency condition necessitates the assignment of a vector entry to a processor that has at least one nonzero in the corresponding row or column of the matrix. In other words, since the vector entries are associated with the nets [4, 6], the consistency condition necessitates the assignment of the vector entry associated with a net n_i to a processor corresponding to a part in $\Lambda(i)$. In the unsymmetric partitioning case, the consistency condition is easily satisfied since the input and output vectors are partitioned independently. In the symmetric partitioning case, the consistency condition is usually satisfied by modifying the sparsity pattern of the matrix to have a zero-free diagonal [4, 6, 13] and then applying hypergraph partitioning to the modified matrix. Designating the owner of a diagonal nonzero as the owner of the corresponding entries in the input and output vectors satisfies the consistency condition in the implicit vector partitioning techniques [4, 6]. This scheme also forms a possible solution in the explicit vector partitioning techniques [2, 13].

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

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 and 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 *elementary 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, 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 $n_y(i) = \{y_i, r_i\}$. 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.

Observe that in the above construction, each net contains a unique vertex that corresponds to either an input vector entry or an output vector entry, i.e., x_i or y_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, because by partitioning the vertices of this hypergraph we can obtain partitions on all operands of a matrix-vector multiply operation.

Figure 3.1(a) and (b) show the data associated with a sample matrix-vector multiply operation and the corresponding elementary hypergraph. In the figure, row 5 has two nonzeros: one in column 4 and another in column 6. Hence, the row vertex r_5 is connected to the nets $n_y(5)$, $n_x(4)$, and $n_x(6)$.

We show how to modify the elementary hypergraph by applying the vertex amalgamation operation to devise 1D unsymmetric and symmetric partitioning models. First, we can apply the owner-computes rule, i.e., y_i should be computed by the

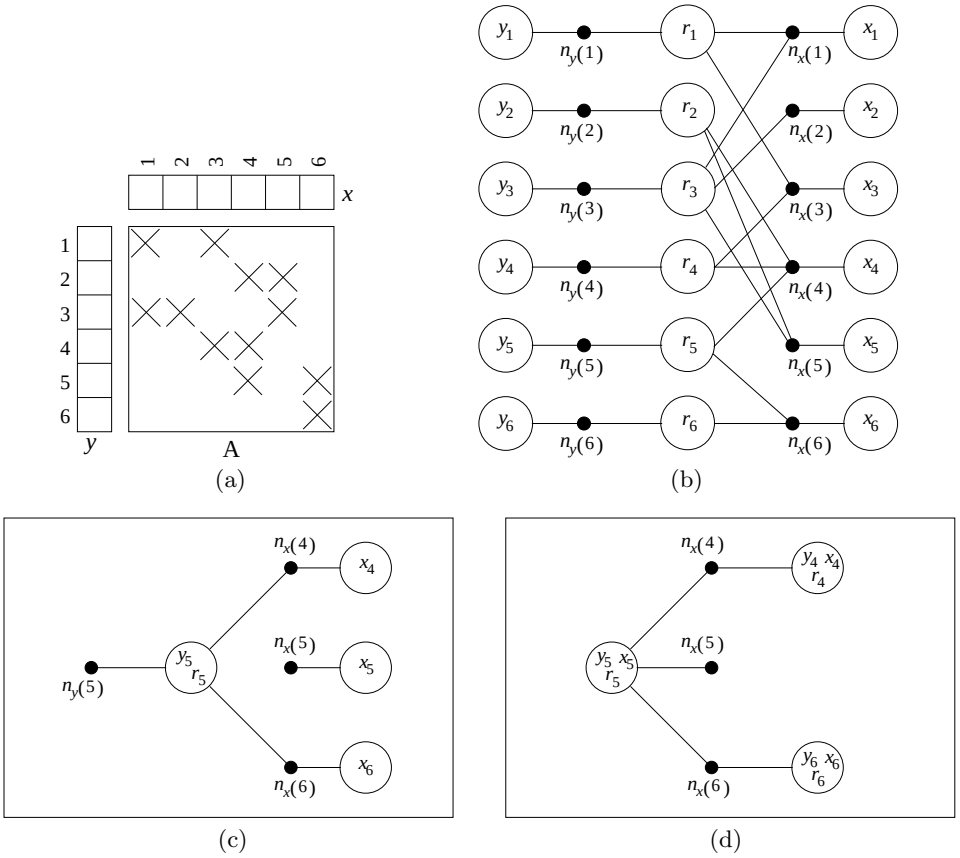

FIG. 3.1. *(a) The operands of a matrix-vector multiply operation with a* 6×6 *matrix* A *and* 6×1 *vectors* x *and* y*. (b) The elementary hypergraph model for 1D partitioning—all operands of the matrix-vector multiply operation are represented by vertices. (c) A portion of the 1D unsymmetric partitioning model—obtained by applying the vertex amalgamation operation to* y_5 *and* r_5 *to enforce the owner-computes rule. (d) A portion of the 1D symmetric partitioning model—obtained by applying the vertex amalgamation operation to the composite vertex* $\langle y_5, r_5 \rangle$ *and the vertex* x_5*.*

processor that owns r_i. This requires amalgamating the vertices y_i and r_i for all i. Figure 3.1(c) shows the amalgamation operation applied to the vertices y_5 and r_5 of the elementary hypergraph given in Fig 3.1(b). Note that after this amalgamation, the size of the net $n_y(i)$, for all i, becomes one. Since the nets of size one do not contribute to the cutsize, we can delete the nets $n_y(i)$ for all i from the model. Partitioning the resulting hypergraph will produce unsymmetric partitions, as the vector entries x_i and y_i might be assigned to different processors.

Suppose we are seeking symmetric partitions; the processor which owns y_i and r_i should own x_i. This time, we have to amalgamate the vertices $\langle y_i, r_i \rangle$ and x_i for all i. Figure 3.1(d) shows the amalgamation operation applied to vertices $\langle y_5, r_5 \rangle$ and x_5 of the model given in Fig 3.1(c). Partitioning the resulting hypergraph will produce symmetric partitions. Note that the hypergraph obtained after these amalgamation operations is structurally equivalent to the column-net hypergraph model proposed in [4] under the zero-free diagonal assumption. However, there is a difference in the semantics. The x-vector entries are represented by the vertices in this work, whereas

they are represented by the nets in [4]. Note that this association guarantees $v_i \in n_i$ for all i independent of the sparsity pattern of the matrix. This justifies enforcing zero-free diagonals in the symmetric partitioning models proposed in [4].

Assume we have partitioned the data of $y \leftarrow Ax$ among K processors by partitioning the unsymmetric or symmetric partitioning models into K parts. In both cases, if the vertex associated with x_j is in $\mathcal{V}_k$, e.g., $x_j \in \mathcal{V}_k$ or $\langle x_j, y_j, r_j \rangle \in \mathcal{V}_k$, then the processor P_k will send x_j to the processors corresponding to the parts in the connectivity set $\Lambda_x(j)$ of $n_x(j)$. In other words, the cutsize accurately represents the total communication volume without any condition. This is true even if the processor that holds x_j has no nonzeros in column j of the matrix. Consider a 6-way partitioning of the hypergraph model for unsymmetric partitioning of the data given in Fig. 3.1(a) in which processor P_i, for $i = 1, \ldots, 6$, gets the composite vertex $\langle y_i, r_i \rangle$ and the x-vertex x_i. Now, observe that P_5 has no nonzeros in column 5 and the communication volume regarding x_5 is $\lambda_x(5) - 1 = 3 - 1 = 2$.

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. As usual, we assume that there are K processors, and the data associated with each part of the K-way vertex partition are assigned to a distinct processor.

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 §3 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 fixed 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 that $\lambda_y(i) - 1$ 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 $y \leftarrow Ax$ computations should be carried out by the parallel algorithm given for Problem 1. We take the elementary hypergraph model given in §3 (see Fig 4.1(a)) and then amalgamate the vertices x_i and y_i into a single vertex. A portion of the resulting hypergraph is shown in Fig. 4.1(b). 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 $\langle x_i, y_i \rangle$ 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 $x \leftarrow x + y$.*

We start with the elementary hypergraph model for $y \leftarrow Ax$ given in §3 (see

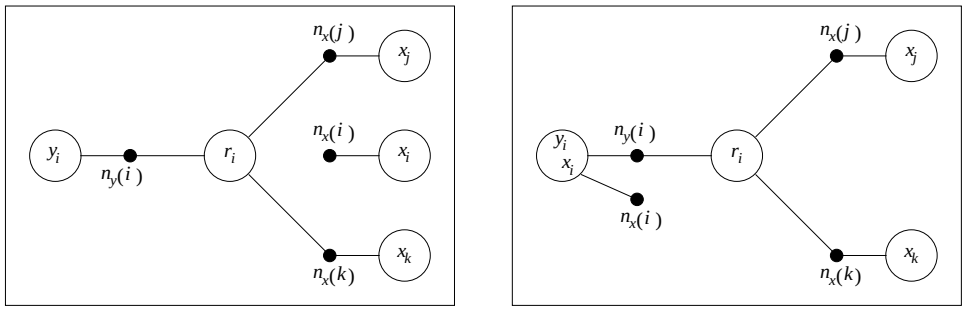

FIG. 4.1. *(a) Portion of the elementary hypergraph model for $y \leftarrow Ax$ with a hypothetical matrix A. The ith row of A is assumed to have two nonzeros: one in column j and another in column k. (b) Hypergraph model for the partitioning problem 2.*

Fig. 4.2(a)). The $x_i + y_i$ addition operations 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 $n_y(i)$. Furthermore, since x_i is dependent on the vertex $x_i + y_i$ due to the computation $x_i \leftarrow x_i + y_i$, we create a new net $n_{x+y}(i)$ and connect x_i to $n_{x+y}(i)$. A portion of the hypergraph with the new vertices representing the $x_i \leftarrow x_i + y_i$ computations and the new nets encoding the dependencies inherent in those computations is shown in Fig. 4.2(b). First, we enforce the owner-computes rule for the $x_i \leftarrow x_i + y_i$ computations. 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(c). Next, we enforce the owner-computes rule for y_i by amalgamating vertices y_i and r_i (Fig. 4.2(d)). In order to carry out the $x_i \leftarrow x_i + y_i$ computations, the y_i values should be communicated after computing $y \leftarrow Ax$. Here, if the composite vertex $\langle x_i, x_i + y_i \rangle$ and the composite vertex $\langle y_i, r_i \rangle$ 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 the owner-computes rule for the y-vector entries is not a must. The hypergraph in Fig. 4.2(c) can be used to address this partitioning problem. Here, if $\langle x_i, x_i + y_i \rangle$, y_i, and r_i reside in different processors, then we will have two units of communication: the result of the inner product $r_i^T \cdot x$ 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, the composite vertex $\langle x_i, x_i + y_i \rangle$ and r_i reside in the same processor, we will have one unit of communication: the result of $r_i^T \cdot x$ 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 $y_i = r_i^T \cdot x$. Similarly, if $\langle x_i, x_i + y_i \rangle$ and y_i reside in the same processor, we will have one unit of communication: the result of $r_i^T \cdot x$ will be sent to that processor which in turn will update y_i and perform $x_i \leftarrow x_i + y_i$.

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 simultaneously. We showed how to transform the elementary model into hypergraph

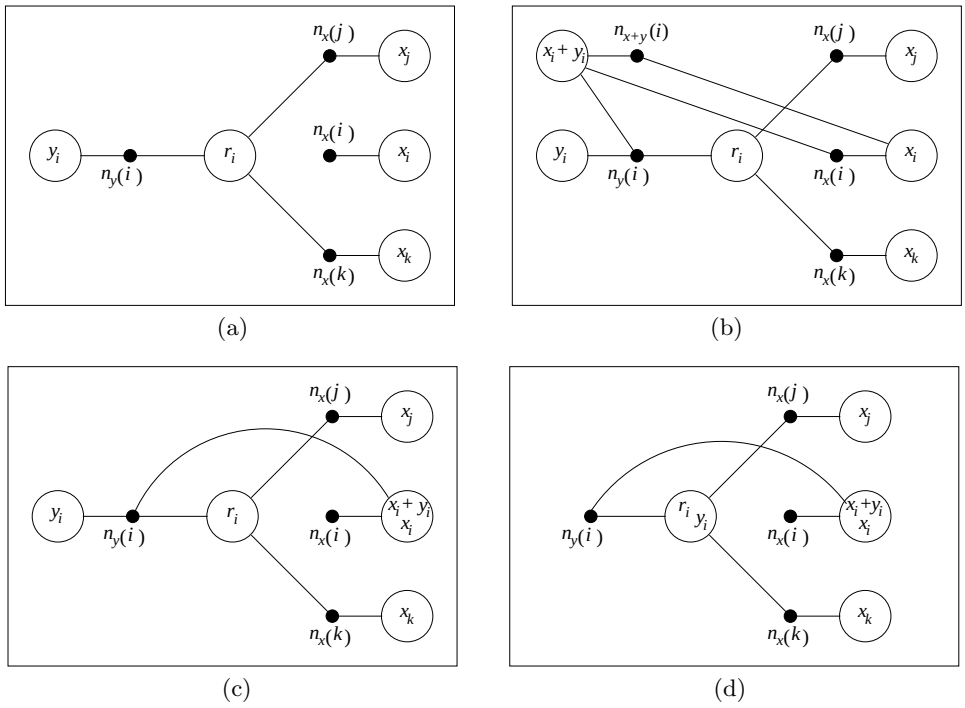

FIG. 4.2. *(a) Portion of the elementary hypergraph model for $y \leftarrow Ax$ with a hypothetical matrix A. The ith row of A is assumed to have two nonzeros: one in column j and another in column k. (b) Initial hypergraph model for the partitioning problem 3 obtained by incorporating new vertices representing the $x_i \leftarrow x_i + y_i$ computations and new nets encoding the dependencies inherent in those computations. (c) According to the owner-computes rule for the $x_i \leftarrow x_i + y_i$ computations, the vertices x_i and $x_i + y_i$ are amalgamated. (d) According to the owner-computes rule for y_i, the vertices y_i and r_i are amalgamated.*

models that can be used to address various 1D partitioning problems including the symmetric and unsymmetric partitioning problems. Although the latter two problems are well studied, the models discussed here shed light on 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; $\mathcal{N} = \mathcal{N}_x \cup \mathcal{N}_y$ 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 $n_y(i) = \{c_j : j = 1, \ldots, n \text{ and } a_{ij} \neq 0\} \cup \{y_i\}$.

The basic ideas can be carried over to the fine-grain partitioning model [6]—two-dimensional, nonzero-based—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 the vertices x_i and y_i for $1 \leq i \leq n$ (if the matrix is $n \times n$) yields a model, whose partitioning results in symmetric partitioning.

Consider a partition of the model H_{2D} for symmetric partitioning, e.g., after the vertex amalgamation operations. The cutsize corresponds exactly to the total communication volume, i.e., the model satisfies the consistency condition. The composite vertex $\langle x_i, y_i \rangle$ is in the nets $n_x(i)$ and $n_y(i)$. Therefore, the *connectivity*-1 value of the nets $n_x(i)$ and $n_y(i)$ again corresponds to the volume of communication regarding x_i and y_i, respectively. That is, if the composite vertex $\langle x_i, y_i \rangle \in \mathcal{V}_k$, then the processor P_k will send x_i to the processors corresponding to the parts in $\Lambda_x(i)$ and will receive the contributions to y_i from the processors corresponding to the parts in $\Lambda_y(i)$. Since the part $\mathcal{V}_k$ is also in both $\Lambda_x(i)$ and $\Lambda_y(i)$, the volume of communications regarding x_i and y_i are $\lambda_x(i) - 1$ and $\lambda_y(i) - 1$, respectively. 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 $\langle x_i, y_i \rangle$. If a_{ii} is zero in A, then the vertex $\langle x_i, y_i \rangle$ can be said to be equivalent to the dummy vertex d_{ii}. If, however, a_{ii} is nonzero in A, then the vertex $\langle x_i, y_i \rangle$ 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 performance of the models in terms of the cutsize (total communication volume).

We should mention that the owner-computes rule should be enforced for two reasons, unless otherwise dictated by the problem. First, it reduces the number of vertices and possibly the number of nets, leading to a reduction in the model size and in the running time of the partitioning algorithm. Second, it avoids a communication phase in the parallel algorithms.

The 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, they conceal the freedom in assigning vector entries to processors to optimize other cost metrics. For example, the vertex x_5 in Fig. 3.1(c) can be re-assigned to any processor in $\Lambda_x(5)$ without changing the computational loads of the processors to reduce communication cost (see [2, 10, 11, 13]).

Acknowledgment. We thank Prof. R. Bisseling of Utrecht University for helpful suggestions on the paper and anonymous referees for their constructive suggestions on the presentation.

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] Ü. V. Çatalyürek and C. Aykanat, *Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication*, IEEE Transactions Parallel and Distributed Systems, 10 (1999), pp. 673–693.
[5] Ü. V. Çatalyürek and C. Aykanat, *PaToH: A multilevel hypergraph partitioning tool, version 3.0*, Tech. Rep. BU-CE-9915, Computer Engineering Department, Bilkent University, 1999.
[6] Ü. V. Çatalyürek and C. Aykanat, *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] B. Uçar and C. Aykanat, *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] B. Uçar and C. Aykanat, *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.