Reduce Operations: Send Volume Balancing While Minimizing Latency

M. Ozan Karsavuran, Seher Acer, and Cevdet Aykanat

Abstract—Communication hypergraph model was proposed in a two-phase setting for encapsulating multiple communication cost metrics (bandwidth and latency), which are proven to be important in parallelizing irregular applications. In the first phase, computational-task-to-processor assignment is performed with the objective of minimizing total volume while maintaining computational load balance. In the second phase, communication-task-to-processor assignment is performed with the objective of minimizing total number of messages while maintaining communication-volume balance. The reduce-communication hypergraph model suffers from failing to correctly encapsulate send-volume balancing. We propose a novel vertex weighting scheme that enables part weights to correctly encode send-volume loads of processors for send-volume balancing. The model also suffers from increasing the total communication volume during partitioning. To decrease this increase, we propose a method that utilizes the recursive bipartitioning framework and refines each bipartition by vertex swaps. For performance evaluation, we consider column-parallel SpMV, which is one of the most widely known applications in which the reduce-task assignment problem arises. Extensive experiments on 313 matrices show that, compared to the existing model, the proposed models achieve considerable improvements in all communication cost metrics. These improvements lead to an average decrease of 30 percent in parallel SpMV time on 512 processors for 70 matrices with high irregularity.

Index Terms—Communication hypergraph, communication cost, maximum communication volume, communication volume, latency, recursive bipartitioning, hypergraph partitioning, sparse matrix, sparse matrix-vector multiplication

1 Introduction

Several successful partitioning models and methods have been proposed for efficient parallelization of irregular applications on distributed memory systems. These partitioning models and methods aim at reducing communication overhead while maintaining computational load balance [1], [2], [3], [4], [5], [6]. Encapsulating multiple communication cost metrics is proven to be important in reducing communication overhead for scaling irregular applications [7], [8], [9], [10], [11], [12], [13], [14], [15].

The communication hypergraph model was proposed for modeling the minimization of multiple communication cost metrics in a two-phase setting [7], [8], [9], [10]. This model was first proposed by Uçar and Aykanat [7] for parallel sparse matrix-vector multiplication (SpMV) based on one-dimensional (1D) partitioning of sparse matrices. Later, this model was extended for two-dimensional (2D) fine-grain partitioned sparse matrices [8] and 2D-checkerboard and 2D-jagged partitioned sparse matrices [9]. Communication hypergraph models were also developed for parallel sparse matrix-matrix multiplication operations based on 1D partitions [10].

- *M.O. Karsavuran and C. Aykanat are with Computer Engineering Department, Bilkent University, Ankara 06800, Turkey. E-mail: {ozan.karsavuran, aykanat}@cs.bilkent.edu.tr.*
- *S. Acer is with the Center for Computing Research, Sandia National Laboratories, Albuquerque, NM 87185. E-mail: sacer@sandia.gov.*

Manuscript received 7 Feb. 2019; revised 5 Nov. 2019; accepted 31 Dec. 2019. Date of publication 7 Jan. 2020; date of current version 11 Feb. 2020. (Corresponding author: Cevdet Aykanat.) Recommended for acceptance by P. Sadayappan. Digital Object Identifier no. 10.1109/TPDS.2020.2964536

The communication hypergraph model encapsulates multiple communication cost metrics in a two-phase setting as follows. In the first phase, computational-task-to-processor assignment is performed with the objective of minimizing total communication volume while maintaining computational load balance. Several successful graph/hypergraph models and methods are proposed for the first phase [1], [3], [5], [8], [10], [16], [17]. In the second phase, communication-task-to-processor assignment is performed with the objective of minimizing total number of messages while maintaining communication-volume balance. The computational-task-to-processor assignment obtained in the first phase determines the communication tasks to be distributed in the second phase. The communication hypergraph model was proposed for assigning these communication tasks to processors in the second phase.

In the communication hypergraph model, vertices represent communication tasks (expand and/or reduce tasks) and hyperedges (nets) represent processors where each net is anchored to the respective part/processor via a fixed vertex. The partitioning objective of minimizing the cutsize correctly encapsulates minimizing the total number of messages, i.e., total latency cost.

In this model, the partitioning constraint of maintaining balance on the part weights aims to encode maintaining balance on the communication volume loads of the processors. Communication volume balancing is expected to decrease the communication load of the maximally loaded processor. The communication volume load of a processor is considered as its send-volume load, whereas receive-volume load is omitted with the assumption that each processor has

enough local computation that overlaps with incoming messages in the network [7], [8], [9], [10].

An accurate vertex weighting scheme is needed for part weights to encode send-volume loads of processors. The vertex weighting scheme proposed for the expand-communication hypergraph enables the part weights to correctly encode the send-volume loads of processors [7]. However, the vertex weighting scheme proposed for the reduce-communication hypergraph fails to encode send-volume loads of processors as already reported in [7]. The authors of [7] explicitly stated that their partitioning constraint corresponds to an approximate the send-volume load balancing and report this approximation to be a reasonable one only if net degrees are close to each other.

In this work, in order to address the above-mentioned deficiency of the reduce-communication hypergraph model, we propose a novel vertex weighting scheme so that a part weight becomes exactly equal to the send-volume load of the respective processor. The proposed vertex weighting scheme involves negative vertex weights. Since the current implementations of hypergraph partitioning tools do not support negative vertex weights, we propose a vertex reweighting scheme to transform all vertex weights to positive values.

The communication hypergraph models also suffer from outcast vertex assignment. In a partition, a vertex assigned to a part is said to be outcast if it is not connected by the net anchored to that part. Outcast vertices have the following adverse effects: First, communication volume increases with increasing number of outcast vertices, so that balancing the communication volume loads of processors begins to loosely relate to minimizing the maximum communication volume load. Second, the correctness of the proposed vertex weighting scheme may decrease with increasing number of outcast vertices. So the number of outcast vertices should be reduced as much as possible to avoid these adverse effects.

In this work, we also propose a method for decreasing the number of outcast vertices during the partitioning of the reduce-communication hypergraph. The proposed method utilizes the well known recursive bipartitioning (RB) framework. After each RB step, the proposed method refines the bipartition by swapping outcast vertices so that they are not outcast anymore. This method involves swapping as many outcast vertices as possible without increasing the cutsize and without disturbing the balance of the current bipartition.

For evaluating the performance of the proposed models, we consider column-parallel SpMV, which is one of the most widely known applications in which the reduce-task assignment problem arises. We conduct extensive experiments on the reduce-communication hypergraphs obtained from 1D column-wise partitioning of 313 sparse matrices. The performance of the proposed models are reported and discussed both in terms of multiple communication cost metrics attained for column-parallel SpMV as well as runtime of column-parallel SpMV on a distributed memory system. Compared to the baseline model, the proposed model achieves an average improvement of 30 percent in parallel SpMV time on 512 processors for 70 matrices that have high level of irregularity.

The rest of the paper is organized as follows: Section 2 defines the reduce-task assignment problem, gives the background material on the reduce-communication hypergraph model, and then explains its above-mentioned deficiencies. The proposed vertex weighting scheme and outcast vertex elimination algorithm are described in Section 3. Section 4 presents experiments, and Section 5 concludes.

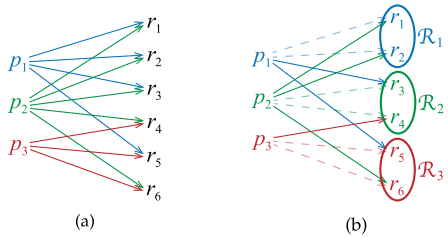

Fig. 1. (a) Three processors, six reduce tasks and partial results in between and (b) A partition of reduce tasks in (a) ($\mathcal{R}_k$ assigned to p_k).

2 Communication Hypergraph for Reduce Operations

2.1 Reduce-Task Assignment Problem

Assume that the target application to be parallelized involves computational tasks that produce partial results for possibly multiple data elements. Also assume that computational-task-to-processor assignment has already been determined in the first phase. Based on this assignment, if there are at least two processors that produce a partial result for an output data element, then those results are reduced to obtain a final value through communication. Here and hereinafter, reducing the partial results to a final value is referred to as a reduce-task. Each reduce-task is assigned to a processor, which is the sole processor that holds the final value of the respective output data element.

Let $\mathcal{R} = \{r_1, r_2, \ldots, r_n\}$ denote the set of reduce tasks for which at least two processors produce a partial result. Let $results(p_k) \subseteq \mathcal{R}$ denote the set of reduce tasks for which processor p_k produces a partial result. Fig. 1a illustrates three processors, six reduce tasks and the partial results in between. For example, p_3 computes partial results for reduce tasks r_4, r_5, and r_6, that is $results(p_3) = \{r_4, r_5, r_6\}$. Reduce task r_6 needs partial results from p_2 and p_3.

Let $\Pi = \{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_K\}$ denote a K-way partition of reduce tasks for a K-processor system, where reduce tasks in $\mathcal{R}_k$ are assumed to be assigned to processor p_k. Then p_k needs to send the partial results in $results(p_k) - \mathcal{R}_k$ to the processors to which respective reduce tasks are assigned.

In the reduce-task partition Π, the amount of data sent by p_k, i.e., communication volume load of p_k, is defined as

$$vol^{\Pi}(p_k) = |results(p_k) - \mathcal{R}_k|, \tag{1}$$

in terms of words. Then, the maximum volume of communication handled by processors becomes

$$vol^{\Pi}_{max} = \max_k vol^{\Pi}(p_k). \tag{2}$$

In the reduce-task partition Π, the number of messages sent by p_k, i.e., latency cost of p_k, is

$$nmsg^{\Pi}(p_k) = |\{\mathcal{R}_{m \neq k} | results(p_k) \cap \mathcal{R}_m \neq \emptyset\}|. \tag{3}$$

That is, $nmsg^{\Pi}(p_k)$ is equal to the number of distinct processors to which the reduce tasks in $results(p_k) - \mathcal{R}_k$ are

assigned. Then, the total number of messages, i.e., total latency cost, becomes

$$nmsg^{\Pi}_{tot} = \sum_{k=1}^{K} nmsg^{\Pi}(p_k). \tag{4}$$

Definition 1 The Reduce-Task Assignment Problem. *Consider a set of reduce tasks $\mathcal{R} = \{r_1, r_2, \ldots, r_n\}$. Assume that $results(p_k) \subseteq \mathcal{R}$ is given for $k = 1, 2, \ldots, K$. Reduce-task assignment problem is defined as the problem of finding a K-way partition $\Pi = \{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_K\}$ of $\mathcal{R}$ with the objective of minimizing both vol^{Π}_{max} and $nmsg^{\Pi}(p_k)$ given in (2) and (4), respectively.*

Fig. 1b illustrates a 3-way partition Π of the reduce tasks displayed in Fig. 1a. In the figure, the set of reduce-tasks in $\mathcal{R}_k$ is assigned to processor p_k for $k = 1, 2, 3$. A dashed arrow line denotes a processor producing a result for a local reduce task, whereas a solid arrow line denotes a processor producing a result for a reduce task assigned to another processor. So, dashed arrow lines do not incur communication whereas solid arrow lines incur communication. In Π, $vol^{\Pi}(p_2) = |results(p_2) - \mathcal{R}_2| = |\{r_1, r_2, r_3, r_4, \ r_6\} - \{r_3, r_4\}| = 3$. Similarly $vol^{\Pi}(p_1) = 2$ and $vol^{\Pi}(p_3) = 1$. Then, $vol^{\Pi}_{max} = vol^{\Pi}(p_2) = 3$. In Π, $nmsg^{\Pi}(p_2) = |\{\mathcal{R}_1, \mathcal{R}_3\}| = 2$. Similarly $nmsg^{\Pi}(p_1) = 2$ and $nmsg^{\Pi}(p_3) = 1$. Then, $nmsg^{\Pi}_{tot} = 2 + 2 + 1 = 5$.

2.2 Reduce-Communication Hypergraph Model

2.2.1 Hypergraph Partitioning (HP) Problem

A hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{N})$ is defined as the set $\mathcal{V}$ of vertices and set $\mathcal{N}$ of nets. Each net n connects a subset of vertices, which is denoted by $Pins(n)$. In $\mathcal{H}$, each vertex v is assigned a weight $w(v)$ and each net n is assigned a cost $c(n)$.

$\Pi = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K\}$ denotes a K-way partition of the vertices in hypergraph $\mathcal{H}$. Let $\lambda(n)$ denote the number of parts that net n connects in Π. Net n is called a cut net if it connects at least two parts, i.e., $\lambda(n) > 1$, and internal (uncut) otherwise. In Π, the weight of part $\mathcal{V}_k$ is defined as

$$W(\mathcal{V}_k) = \sum_{v \in \mathcal{V}_k} w(v). \tag{5}$$

In the HP problem, the partitioning objective is to minimize the connectivity cutsize [1] which is defined as

$$cutsize(\Pi) = \sum_{n \in \mathcal{N}} (\lambda(n) - 1) c(n), \tag{6}$$

and the partitioning constraint is to satisfy the constraint

$$W(\mathcal{V}_k) \leq W_{avg}(1 + \epsilon), \tag{7}$$

for each part $\mathcal{V}_k$ in Π, for a given maximum allowed imbalance ratio ϵ. Here W_{avg} denotes the weight of each part under perfect balance, that is,

$$W_{avg} = \frac{W_{tot}}{K}, \text{ where } W_{tot} = \sum_{k=1}^{K} W(\mathcal{V}_k). \tag{8}$$

Note that the total vertex weight W_{tot} is constant and does not change with different partitions. Hence, the partitioning constraint of maintaining balance on the part weights (7) by utilizing sufficently small ϵ values corresponds to minimizing the maximum part weight.

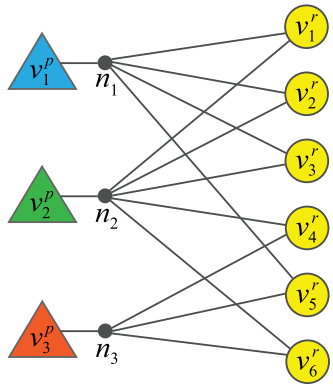

Fig. 2. Reduce-communication hypergraph of the reduce-task assignment problem given in Fig. 1a.

The HP problem with fixed vertices is a version of the HP problem in which the assignments of some vertices are determined before partitioning. These vertices are called fixed vertices and $\mathcal{F}_k$ denotes the set of vertices that are fixed to part $\mathcal{V}_k$. At the end of the partitioning, vertices in $\mathcal{F}_k$ remain in $\mathcal{V}_k$, i.e., $\mathcal{F}_k \subseteq \mathcal{V}_k$. The rest of the vertices are called free vertices.

2.2.2 Reduce-Communication Hypergraph Model

The reduce-communication hypergraph model [7] $\mathcal{H} = (\mathcal{V}^p \cup \mathcal{V}^r, \mathcal{N})$ contains two types of vertices, which correspond to processors and reduce tasks, and a single net type, which corresponds to processors. Each processor p_k is represented by a vertex v^p_k in $\mathcal{V}^p$, whereas each reduce task r_i in $\mathcal{R}$ is represented by a vertex v^r_i in $\mathcal{V}^r$. Then the set of vertices $\mathcal{V}$ is formulated by

$$\mathcal{V} = \mathcal{V}^p \cup V^r = \{v^p_1, v^p_2, \ldots v^p_K\} \cup \{v^r_i : r_i \in \mathcal{R}\}. \tag{9}$$

Each processor p_k is also represented by a net n_k in $\mathcal{N}$. Then the set of nets $\mathcal{N}$ is formulated by

$$\mathcal{N} = \{n_1, n_2, \ldots, n_K\}. \tag{10}$$

Each net n_k connects the vertex that represents p_k as well as the vertices that represent the reduce tasks for which p_k produces a partial result. That is,

$$Pins(n_k) = \{v^p_k\} \cup \{v^r_i : r_i \in results(p_k)\}. \tag{11}$$

The vertices in $\mathcal{V}^p$ are assigned zero weight, whereas the vertices in $\mathcal{V}^r$ are assigned unit weight. That is,

$$\begin{aligned} w(v^p_k) &= 0, \forall v^p_k \in \mathcal{V}^p \\ w(v^r_i) &= 1, \forall v^r_i \in \mathcal{V}^r. \end{aligned} \tag{12}$$

The nets in $\mathcal{N}$ are assigned unit cost. That is,

$$c(n_k) = 1, \forall n_k \in \mathcal{N}. \tag{13}$$

The reduce-communication hypergraph model utilizes fixed vertices. The vertices in $\mathcal{V}^p$ are fixed, whereas the vertices in $\mathcal{V}^r$ are free. For $k = 1, 2, \ldots, K$, vertex v^p_k in $\mathcal{V}^p$ is fixed to part $\mathcal{V}_k$, i.e., $\mathcal{F}_k = \{v^p_k\}$.

Fig. 2 displays the reduce-communication hypergraph of the reduce-task assignment problem shown in Fig. 1a. In the figure, fixed and free vertices are represented by triangles and

circles, respectively. Nets and pins are represented by small circles and lines, respectively.

A K-way partition

$$\Pi = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K\}, \quad \text{where } \mathcal{V}_k = \{v_k^p\} \cup \mathcal{V}_k^r,$$

of reduce-communication hypergraph $\mathcal{H}$ is decoded as follows. Each free vertex v_i^r in $\mathcal{V}_k$ induces that the reduce task r_i is assigned to processor p_k since $v_k^p \in \mathcal{V}_k$ in Π. That is, vertex partition $\{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K\}$ induces a reduce-task partitioning $\{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_K\}$, where $\mathcal{R}_k$ contains the reduce-tasks corresponding to the vertices in $\mathcal{V}_k^r$. So we use the symbol Π for both reduce-task partition/assignment and hypergraph partition interchangeably.

The partitioning objective of minimizing the cutsize (6) encodes minimizing the number of messages, which is also referred to as the latency cost (4). During partitioning communication hypergraphs, almost all nets remain cut. This is because a communication hypergraph contains small number (as many as the number of processors/parts) of nets with possibly high degrees and an uncut net refers to a processor that does not send any messages. Therefore it is important to utilize the connectivity cutsize metric in (6). The vertex weight definition given in (12) encodes the part weight (5) as the number of reduce-tasks assigned to the respective processor. So, the partitioning constraint of maintaining balance on the part weights corresponds to maintaining balance on the number of reduce-tasks assigned to processors, i.e., $|\mathcal{R}_k|$ values.

2.3 Deficiencies of Reduce-Communication Hypergraph

2.3.1 Failure to Encode Communication Volume Loads of Processors

The part weights computed according to the vertex weighting scheme utilized in the reduce-communication hypergraph model fails to correctly encapsulate the communication volume loads of processors. That is, the existing reduce-communication hypergraph model computes the volume load of processor p_k as

$$vol^{\Pi}(p_k) = |\mathcal{R}_k|, \tag{14}$$

whereas the actual volume load of p_k is

$$vol^{\Pi}(p_k) = |results(p_k) - \mathcal{R}_k|. \tag{15}$$

So, the partitioning constraint of maintaining balance on part weights does not correctly correspond to maintaining communication volume load balancing.

In regular reduce-task assignment instances, processors produce partial results for similar number of reduce tasks, that is, they have similar $|results(p_k)|$ values, which corresponds to similar net degrees. For such regular instances, the approximation provided by the existing reduce-communication hypergraph model can be considered reasonable, as also reported in [7]. This is because the existing reduce-communication hypergraph model makes a similar amount of error in computing the volume loads of processors according to (14), hence maintaining balance on the $|\mathcal{R}_k|$ values corresponds to maintaining balance on the volume loads. However, the deficiency of the existing model in encapsulating correct communication volume balancing increases with increasing irregularity in net degrees.

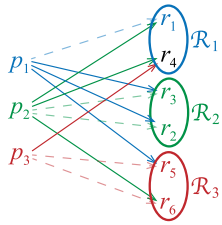

Fig. 3. A partition of reduce-tasks shown in Fig. 1a with an outcast reduce task (r_4).

Fig. 1b exemplifies the above-mentioned deficiency. Note that $|\mathcal{R}_1| = |\mathcal{R}_2| = |\mathcal{R}_3| = 2$ in $\Pi = \{\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3\}$. Assume that the perfect balance on these $|\mathcal{R}_k|$ values is obtained via achieving a perfect balance on part weights in partitioning the reduce-communication hypergraph model. Also note that $|results(p_1)| = 4$, $|results(p_2)| = 5$, and $|results(p_3)| = 3$. The imbalance on these $|results(p_k)|$ values induces an imbalance on $vol^{\Pi}(p_k)$ values as $vol^{\Pi}(p_1) = 4 - 2 = 2$, $vol^{\Pi}(p_2) = 5 - 2 = 3$, and $vol^{\Pi}(p_3) = 3 - 2 = 1$.

2.3.2 Increase in Total Communication Volume

The existing reduce-communication hypergraph model also suffers from the increase in the total communication volume during the partitioning. A reduce task r_i assigned to a processor which does not compute a partial result for r_i is referred to here as an *outcast reduce task*. Each outcast reduce-task assignment increases the total communication volume by one. However, this increase due to the outcast reduce-tasks is controlled neither by the problem formulation given in Section 2.1 nor by the reduce-communication hypergraph model described in Section 2.2.2. This deficiency has an adverse effect on the correspondence between maintaining communication volume balancing and minimizing the maximum communication volume in the communication hypergraph model. The more the increase in the total communication volume is, the more the above-mentioned adverse effect becomes pronounced.

This is because attaining tight balance on processors' communication volume loads while increasing the total communication volume may not correspond to reducing the maximum communication volume (2).

Fig. 3 displays a reduce-task partition which contains one outcast reduce task. This partition is obtained from the outcast-free partition given in Fig. 1b by changing the assignments of r_2 and r_4 to processors p_2 and p_1, respectively. As seen in the figure, reduce task r_4 is outcast in the current partition since processor p_1 does not compute a partial result for r_4. Note that this change increases the total communication volume by one.

In a K-way partition Π of reduce-communication hypergraph $\mathcal{H}$, we define *outcast vertices* to identify the outcast reduce-task assignments. A vertex v_i^r is called outcast if v_i^r is assigned to a part $\mathcal{V}_k$, where net n_k does not connect v_i^r, that is, $v_i^r \in \mathcal{V}_k$ and $v_i^r \notin Pins(n_k)$. Note that $v_i^r \in \mathcal{V}_k$ signifies that reduce task r_i is assigned to processor p_k and

$v_i^r \notin Pins(n_k)$ signifies that p_k does not compute a partial result for r_i.

In a partition Π, the existence of outcast vertices does not necessarily disturb the partitioning objective of minimizing cutsize (6). Indeed, partitioning the hypergraph while trying to maintain balance without increasing the cutsize might motivate the partitioning tool to assign vertices to parts where they become outcast. Moreover, in the case the partitioning tool discovers a partition with outcast vertices, it has no motivation to refine it as long as the cutsize and the imbalance on the part weights remain the same.

3 Correct Reduce-Communication Hypergraph Model

3.1 A Novel Vertex Weighting Scheme

In order to minimize the maximum communication volume handled by processors, we propose a novel vertex weighting scheme that encapsulates the communication volume loads of processors via part weights.

Consider a K-way outcast-vertex-free partition Π_{of} of a given reduce-communication hypergraph $\mathcal{H} = (\mathcal{V}^p \cup \mathcal{V}^r, \mathcal{N})$. Here and hereafter, we refer to outcast-vertex-free partition shortly as outcast-free partition. Note that in an outcast-free partition each reduce task r_i is assigned to a processor that computes a partial result for r_i. Let $\{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_K\}$ denote the reduce-task partition/assignment induced by Π_{of}. Then the communication volume load of each processor p_k becomes

$$\begin{aligned} vol^{\Pi_{\text{of}}}(p_k) &= |results(p_k) - \mathcal{R}_k| && \text{(16a)}\\ &= |results(p_k)| - |\mathcal{R}_k| && \text{(16b)}\\ &= (|Pins(n_k)| - 1) - |\mathcal{V}_k^r|. && \text{(16c)} \end{aligned}$$

We obtain (16b) from (16a) since $\mathcal{R}_k \subseteq results(p_k)$ for each part $\mathcal{R}_k$ in an outcast-free partition. We obtain (16c) from (16b) by utilizing the hypergraph theoretical view.

According to (16b), for any outcast-free partition, $|results(p_k)|$ is an upper bound on the send volume load of processor p_k and each reduce-task assigned to processor p_k reduces the communication volume load of p_k by one. In other words, each free vertex that is connected by n_k and assigned to part $\mathcal{V}_k$ reduces the volume load of processor p_k by one. So we propose the following vertex weighting scheme

$$\begin{aligned} w(v_k^p) &= |results(p_k)|, \quad && \forall v_k^p \in \mathcal{V}^p \\ w(v_i^r) &= -1, && \forall v_i^r \in \mathcal{V}^r. \end{aligned} \tag{17}$$

Then the weight of part $\mathcal{V}_k$ becomes

$$\begin{aligned} W(\mathcal{V}_k) &= \sum_{v \in \mathcal{V}_k} w(v) && \text{(18a)}\\ &= w(v_k^p) + \sum_{v_i^r \in \mathcal{V}_k^r} w(v_i^r) && \text{(18b)}\\ &= |results(p_k)| + \sum_{v_i^r \in \mathcal{V}_k^r} (-1) && \text{(18c)}\\ &= |results(p_k)| - |\mathcal{V}_k^r| && \text{(18d)}\\ &= |results(p_k)| - |\mathcal{R}_k| && \text{(18e)}\\ &= vol^{\Pi_{\text{of}}}(p_k). && \text{(18f)} \end{aligned}$$

That is, part weight $W(\mathcal{V}_k)$ will correctly encode the volume of data sent by processor p_k.

As seen in (17), the proposed vertex weighting scheme assigns a negative weight to all free vertices. However, current implementations of hypergraph/graph partitioning tools (PaToH [1], hMETIS [18], METIS [19]) do not support negative vertex weights. We propose the following vertex *reweighting* scheme for transforming all vertex weights to positive values.

We first multiply each vertex weight with -1. This scaling transforms the weights of all free vertices to $+1$, while transforming the weight of each fixed vertex v_k^p to a negative value of $-|results(p_k)|$. Then we shift the weights of fixed vertices to positive values by adding the maximum fixed-vertex weight to the weight of all fixed vertices.

That is, after the proposed reweighting scheme, vertex weights become

$$\begin{aligned} \hat{w}(v_k^p) &= -|results(p_k)| + M_{fvw}, \quad && \forall v_k^p \in \mathcal{V}^p, \\ \hat{w}(v_i^r) &= +1, && \forall v_i^r \in \mathcal{V}^r, \end{aligned} \tag{19}$$

where M_{fvw} denotes the maximum fixed vertex weight, i.e., $M_{fvw} = \max_\ell |results(p_\ell)|$.

Under the proposed vertex reweighting scheme, we can compute the weight of part $\mathcal{V}_k$ as

$$\hat{W}(\mathcal{V}_k) = M_{fvw} - vol^{\Pi_{\text{of}}}(p_k), \tag{20}$$

by following the steps of Equation (18) for an outcast-free partition Π_{of}. As seen in (20), the part weights encode send volume loads of processors with the same constant shift amount M_{fvw}.

Note that maintaining balance on the part weights corresponds to maintaining balance on the send-volume loads of processors. Hence, perfect balance on the part weights corresponds to minimizing the maximum send volume (2) which is one of the objectives of the Reduce-Task Assignment Problem.

We present the following theorem to address the validity of the proposed vertex reweigting scheme.

Theorem 1. *Let $(\mathcal{H}, w)$ denote the reduce-communication hypergraph model with the vertex weighting scheme proposed in (17). Let $(\mathcal{H}, \hat{w})$ denote the model with the vertex reweighting scheme proposed in (19). Then Π^* is a perfectly-balanced partition of $(\mathcal{H}, w)$ if and only if it is a perfectly-balanced partition of $(\mathcal{H}, \hat{w})$.*

Proof. We find the relation between the total vertex weights W_{tot} and $\hat{W}_{tot}$ to derive the relation between average part weights W_{avg} and $\hat{W}_{avg}$ for two vertex weighting schemes w and $\hat{w}$, respectively. The derivations of expressions for W_{avg} and $\hat{W}_{avg}$ are important since in a perfectly-balanced partition the weight of each part should be equal to the average part weight by (7), that is, $W(\mathcal{V}_k) = W_{avg}$ and $\hat{W}(\mathcal{V}_k) = \hat{W}_{avg}$ for $k = 1, \ldots, K$.

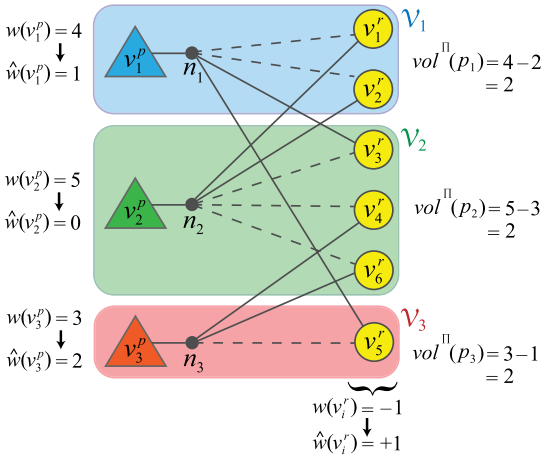

Fig. 4. A balanced partition of the reduce-communication hypergraph given in Fig. 2 with proposed vertex weights.

From (18f) and (20) we obtain

$$W(\mathcal{V}_k) = M_{fvw} - \hat{W}(\mathcal{V}_k). \tag{21}$$

Then, we compute the sum of both sides of (21) for all k

$$\sum_{k=1}^{K} W(\mathcal{V}_k) = \sum_{k=1}^{K} (M_{fvw} - \hat{W}(\mathcal{V}_k)), \tag{22a}$$

$$W_{tot} = KM_{fvw} - \hat{W}_{tot}. \tag{22b}$$

Finally, we divide both sides of (22b) by K to obtain

$$W_{avg} = M_{fvw} - \hat{W}_{avg}. \tag{23}$$

(23) holds because reduce-communication hypergraph model contains K fixed vertices in total, whereas (21) holds because it contains exactly one fixed vertex in each part. Hence shifting the weight of each fixed vertex by M_{fvw}, shifts each part weight and average weight by the same amount M_{fvw}.

$\Rightarrow$ Assume that Π^* is a perfectly-balanced partition of $(\mathcal{H}, w)$. Then, we have

$$W(\mathcal{V}_k) = W_{avg} \quad \text{for } k = 1, \ldots, K. \tag{24}$$

Replacing left hand side by (21) and right hand side by (23), (24) becomes

$$M_{fvw} - \hat{W}(\mathcal{V}_k) = M_{fvw} - \hat{W}_{avg},$$

and hence

$$\hat{W}(\mathcal{V}_k) = \hat{W}_{avg} \quad \text{for } k = 1, \ldots, K.$$

This shows that Π^* is also a perfectly-balanced partition of $(\mathcal{H}, \hat{w})$.

$\Leftarrow$ A dual proof holds. That is, assume $\hat{W}(\mathcal{V}_k) = \hat{W}_{avg}$ and then show $W(\mathcal{V}_k) = W_{avg}$ for $k = 1, \ldots, K$. □

Fig. 4 illustrates a perfectly-balanced partition of the communication hypergraph given in Fig. 2 with vertex weights assigned by the proposed vertex (re)weighting scheme. Dashed pins denote partial results for local reduce tasks, whereas solid pins denote partial results for external ones. So, the number of solid lines (except the one connected to the fixed vertex) incident to each net is equal to the communication volume load of the respective processor As seen in the figure, the weight of fixed vertex v_1^p is $\hat{w}(v_1^p) = -|results(p_1)| + M_{fvw} = -4 + 5 = 1$. Similarly, $\hat{w}(v_2^p) = -5 + 5 = 0$ and $\hat{w}(v_3^p) = -3 + 5 = 2$. As seen in the figure, parts $\mathcal{V}_1$, $\mathcal{V}_2$, and $\mathcal{V}_3$ contain two, three, and one free vertices, respectively. Note that $\hat{W}(\mathcal{V}_1) = 1 + 2 = 3$, $\hat{W}(\mathcal{V}_2) = 0 + 3 = 3$, and $\hat{W}(\mathcal{V}_3) = 2 + 1 = 3$. Also note that $vol^{\Pi^*_{of}}(p_1) = |results(p_1)| - |\mathcal{R}_1| = 4 - 2 = 2$, $vol^{\Pi^*_{of}}(p_2) = |results(p_2)| - |\mathcal{R}_2| = 5 - 3 = 2$, and $vol^{\Pi^*_{of}}(p_3) = |results(p_3)| - |\mathcal{R}_3| = 3 - 1 = 2$.

3.2 Eliminating Outcast Vertices via Recursive Bipartitioning

In this section, we propose a RB-based framework that aims at minimizing the total number of outcast vertices. In this context, we first describe how RB framework works for partitioning communication hypergraphs, which contain one fixed vertex in each part of the resulting K-way partition. Without loss of generality, we assume that the number K of processors is an exact power of 2.

In the RB paradigm, the given hypergraph is bipartitioned into two subhypergraphs, which are further bipartitioned recursively until K parts are obtained. This procedure produces a complete binary tree with $\log_2 K$ levels which is referred as the RB tree. 2^ℓ hypergraphs in the ℓth level of the RB tree are denoted by $\mathcal{H}_0^\ell, \ldots, \mathcal{H}_{2^\ell-1}^\ell$ from left to right for $0 \leq \ell \leq \log_2 K$.

A bipartition $\Pi_2 = \{\mathcal{V}_L, \mathcal{V}_R\}$ of an ℓth level hypergraph $\mathcal{H}_k^\ell$ forms two new vertex-induced subhypergraphs $\mathcal{H}_{2k}^{\ell+1} = (\mathcal{V}_L, \mathcal{N}_L)$ and $\mathcal{H}_{2k+1}^{\ell+1} = (\mathcal{V}_R, \mathcal{N}_R)$, both in level $\ell + 1$. Here, $\mathcal{V}_L$ and $\mathcal{V}_R$ respectively refer to the left and right parts of the bipartition. Internal nets of the left and right parts are assigned to net sets $\mathcal{N}_L$ and $\mathcal{N}_R$ as is, respectively, whereas the cut-nets are assigned to both net sets, only with the pins found in the respective part. That is, $\mathcal{N}_L = \{n_i : Pins(n_i) \cap \mathcal{V}_L \neq \emptyset\}$, where $Pins(n_j^L) = Pins(n_j) \cap \mathcal{V}_L$ for each $n_j^L \in \mathcal{N}_L$, whereas $\mathcal{N}_R$ is formed in a dual manner. This way of forming $\mathcal{N}_L$ and $\mathcal{N}_R$ is known as the cut-net splitting method [1] which is proposed to encode the connectivity cutsize metric (6) in the final K-way partition. Although every net of the reduce-communication hypergraph model connects exactly one fixed vertex, subhypergraphs may contain nets that do not connect a fixed vertex. This stems from the cut-net splitting method described above.

At each RB step, one half of the fixed vertices in the current hypergraph are assigned to $\mathcal{V}_L$, whereas the other half are assigned to $\mathcal{V}_R$, in order to attain one fixed vertex in each part of the final K-way partition. In this way, hypergraph $\mathcal{H}_k^\ell$ contains $K/2^\ell$ fixed vertices.

Algorithm 1 shows the basic steps of the proposed RB-based scheme. In the algorithm, BIPARTITION at line 4 denotes a call to a 2-way HP tool to obtain $\Pi_2 = \{\mathcal{V}_L, \mathcal{V}_R\}$, whereas lines 6 and 7 show the formation of left and right subhypergraphs according to the above-mentioned cut-net splitting technique. The proposed scheme is applied after obtaining bipartition Π_2 through calling SWAP-OUTCAST function at line 5.

In Π_2, a vertex in left part $\mathcal{V}_L$ is said to be outcast if it is not connected by any left-anchored nets. Here, a net is said

Algorithm 1. RB With Swap

Require: $\mathcal{H} = (\mathcal{V}, \mathcal{N})$, K
1: $\mathcal{H}_0^0 = \mathcal{H}$
2: **for** $\ell \leftarrow 0$ **to** $\log_2 K - 1$ **do**
3: **for** $k \leftarrow 0$ **to** $2^\ell - 1$ **do**
4: $\Pi_2 \leftarrow$ BIPARTITION($\mathcal{H}_k^\ell$) ▷ $\Pi_2 = \{\mathcal{V}_L, \mathcal{V}_R\}$
5: $\Pi_2 \leftarrow$ SWAP-OUTCAST($\mathcal{H}_k^\ell, \Pi_2$) ▷ updates Π_2
6: Form $\mathcal{H}_L = \mathcal{H}_{2k}^{\ell+1} = (\mathcal{V}_L, \mathcal{N}_L)$ induced by $\mathcal{V}_L$
7: Form $\mathcal{H}_R = \mathcal{H}_{2k+1}^{\ell+1} = (\mathcal{V}_R, \mathcal{N}_R)$ induced by $\mathcal{V}_R$

to be left-/right-anchored if it connects a fixed vertex in the left/right part. It is clear that an outcast vertex in $\mathcal{V}_L$ remains to be outcast in the further RB steps as well as in the final K-way partition. A similar argument holds for an outcast vertex in $\mathcal{V}_R$. SWAP-OUTCAST function refines Π_2 by swapping outcast vertices so that they are not outcast anymore in Π_2.

The vertices of $\mathcal{V}_L$ satisfying all three condititions given below are defined as candidates for swap operations.

i) v_i^r is not connected by a left-anchored net,
ii) v_i^r is connected by a cut right-anchored net, and
iii) v_i^r is not connected by an internal net in $\mathcal{V}_L$.

The candidate vertices of $\mathcal{V}_R$ are identified in a dual manner.

Condition (i) identifies v_i^r as an outcast vertex of $\mathcal{V}_L$. Condition (ii) ensures that v_i^r would not be outcast in Π_2 if it were assigned to $\mathcal{V}_R$. Thus conditions (i) and (ii) together identify that moving v_i^r to $\mathcal{V}_R$ in a swap operation will make v_i^r not outcast anymore in Π_2. The swap of any two vertices in $\mathcal{V}_L$ and $\mathcal{V}_R$ both of which satisfy conditions (i) and (ii) together reduces the number of outcast vertices in Π_2 by two. The swap operations are preferred over individual moves in order not to disturb the imbalance of the current bipartition.

In a swap operation, moving v_i^r to $\mathcal{V}_R$ increases the cutsize by the number of internal nets that connect v_i^r in $\mathcal{V}_L$. Hence condition (iii) ensures that moving v_i^r to $\mathcal{V}_R$ does not increase the cutsize and thus, the partitioning objective of minimizing the cutsize is not disturbed.

Fig. 5 shows an RB step illustrating different states for vertices in terms of the candidacy for being swapped. For simplicity, we only show them in the left part. Consider vertices v_g^r, v_h^r and v_i^r in $\mathcal{V}_L$. Vertex v_g^r is connected by a left-anchored net (n_a), so, it violates condition (i), which means that it is not outcast. Vertex v_h^r is not connected by a left-anchored net and it is connected by a right-anchored net (n_b). So, it satisfies conditions (i) and (ii), which means that v_h^r would not be outcast in $\mathcal{V}_R$. However, since it is connected by an internal net (n_t), moving it to $\mathcal{V}_R$ increases the cutsize, hence, it violates condition (iii). Vertex v_i^r, on the other hand, satisfies all three conditions, hence, it is a candidate for being swapped.

Algorithm 2 shows the basic steps of the SWAP-OUTCAST algorithm. As seen in the algorithm, the for loop in lines 3–13 makes a single pass over all nets of the current hypergraph $\mathcal{H}$. Lines 4–8 identify the vertices that do not satisfy condition (i) so that candidate flags of these vertices are set to false. Else-If statement in lines 9–10 identifies the vertices that satisfy both conditions (i) and (ii). Line 9 ensures that a vertex is never considered again if it was once found to violate condition (i). Else-If statement in lines 11–13 identifies vertices that do not satisfy condition (iii). Note that a vertex which was found to be

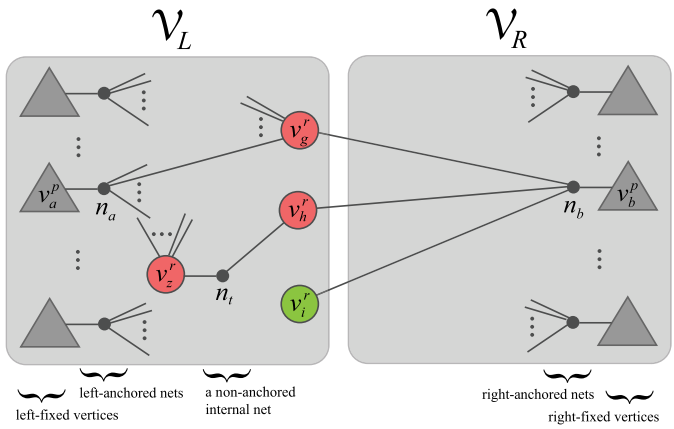

Fig. 5. Among vertices v_g^r, v_h^r, v_i^r, only v_i^r is candidate although both v_h^r and v_i^r are outcast vertices.

candidate earlier can turn out to be violating condition (iii) later. After executing the for loop in lines 3–13, only the vertices that satisfy all three conditions have their candidate flags set to true.

The for loop in lines 15–20 performs a pass over all free vertices to construct the set of swappable vertex sets S_L and S_R by utilizing candidate vertex flags. Finally the while loop in lines 21–26 performs $\min\{|S_L|, |S_L|\}$ swaps.

Algorithm 2. SWAP-OUTCAST

Require: $\mathcal{H} = (\mathcal{V}, \mathcal{N})$, $\Pi_2 = \{\mathcal{V}_L, \mathcal{V}_R\}$
1: **for** each free vertex $v_i^r \in \mathcal{V}$ **do**
2: $cand(v_i^r) \leftarrow$ **maybe**
3: **for** each net $n \in \mathcal{N}$ **do**
4: **if** n connects a fixed vertex **then** ▷ n is an anchored net
5: $v_k^p \leftarrow$ the fixed vertex in $Pins(n)$
6: **for** each free vertex $v_i^r \in Pins(n)$ **do**
7: **if** $part(v_k^p) = part(v_i^p)$ **then**
8: $cand(v_i^r) \leftarrow$ **false**
9: **else if** $cand(v_i^r) \neq$ **false then**
10: $cand(v_i^r) \leftarrow$ **true**
11: **else if** n is internal **then**
12: **for** each $v_i^r \in Pins(n)$ **do**
13: $cand(v_i^r) \leftarrow$ **false**
14: $S_L \leftarrow \emptyset$ and $S_R \leftarrow \emptyset$ ▷ swappable Left/Right vertex sets
15: **for** each free vertex $v_i^r \in \mathcal{V}$ **do**
16: **if** $cand(v_i^r) =$ **true then**
17: **if** $part(v_i^r) = L$ **then**
18: $S_L \leftarrow S_L \cup \{v_i^r\}$
19: **else**
20: $S_R \leftarrow S_R \cup \{v_i^r\}$
21: **while** $S_L \neq \emptyset$ and $S_R \neq \emptyset$ **do** ▷ $\min\{|S_L|, |S_L|\}$ swaps
22: Let $v_i^r \in S_L$ and $v_j^r \in S_R$
23: $part(v_i^r) \leftarrow R$ ▷ swap v_i^r and v_j^r
24: $part(v_j^r) \leftarrow L$
25: $S_L \leftarrow S_L - \{v_i^r\}$
26: $S_R \leftarrow S_R - \{v_j^r\}$

*Here $cand$ refers to a three-state variable, where true denotes swappable, false denotes not swappable and maybe denotes not decided yet.

The running time of the proposed SWAP-OUTCAST algorithm is $\Theta(P + |\mathcal{V}|)$, where P denotes the total number of pins in $\mathcal{H}$. Note that proposed SWAP-OUTCAST

algorithm is quite efficient since it performs a single pass over pins and free vertices of $\mathcal{H}$.

4 Experiments

4.1 Test Application: Column-Parallel SpMV

SpMV is denoted by $y \leftarrow Ax$, where $A = (a_{ij})$ is an $n \times m$ sparse matrix and $x = (x_i)$ and $y = (y_j)$ are dense vectors. In column-parallel SpMV, the columns of matrix A are distributed among processors as well as the entries of vectors x and y. The partitions of columns of A and entries of x and y are obtained by a two-phase partitioning approach.

In the first phase, the row-net hypergraph partitioning model [1] is utilized to obtain a partition of columns of A in such a way that the total communication volume is minimized while maintaining balance on the computational loads of processors. This column partition induces a conformable partition on the input vector x, that is, x_i is assigned to the processor to which column i is assigned. Note that assigning all nonzeros of column i together with x_i to a single processor eliminates the need for the pre-communication phase, which is performed for broadcasting x-vector entries. However, since multiple processors may produce partial results for the same y-vector entries, the post-communication phase needs to be performed to reduce those partial results for obtaining final values of y-vector entries.

In the second phase, a partition of output vector y is obtained via the proposed reduce-communication hypergraph model as follows. The set of reduce tasks, $\mathcal{R}$, corresponds to the subset of y-vector entries for which multiple processors compute a partial result. That is,

$$\mathcal{R} = \{r_i : \exists \text{ columns } j_1 \text{ and } j_2 \text{ assigned to different processors and } a_{i,j_1} \neq 0 \text{ and } a_{i,j_2} \neq 0\}.$$

Here, r_i represents the reduce-task associated with y_i, as well as row i. Then, $results(p_k)$ can be formulated as

$$results(p_k) = \{r_i : a_{ij} \neq 0 \text{ and column } j \text{ is assigned to } p_k\}.$$

Note that a row whose nonzeros are all assigned to a single processor does not incur a reduce task.

4.2 Setup

The performance of the proposed models are compared against the existing reduce-communication hypergraph model [7] (Section 2.2.2) which is referred to as the baseline model $\mathrm{RC_b}$. The reduce-communication hypergraph model that utilizes the proposed novel vertex (re)weighting scheme (Section 3.1) is referred to as $\mathrm{RC_{vw}}$, whereas the model that utilizes both the proposed vertex weighting scheme and the proposed outcast vertex elimination scheme (Section 3.2) is referred to as $\mathrm{RC^s_{vw}}$. We used $K = 512$ processors for performance comparison of these models.

We use PaToH [1] for partitioning both row-net hypergraphs and reduce-communication hypergraphs, in the first and second phases, respectively. For the row-net, $\mathrm{RC_b}$, and $\mathrm{RC_{vw}}$ models, PaToH is called for K-way partitioning for a K-processors system, whereas for the $\mathrm{RC^s_{vw}}$ model, PaToH is used for 2-way partitioning (line 4 of Algorithm 1). PaToH is used with default parameters for partitioning row-net hypergraph, whereas refinement algorithm is set to boundary FM for partitioning communication hypergraphs. Maximum allowed imbalance is set to 10 percent, i.e., $\epsilon = 0.10$, for all models. Since PaToH utilizes randomized algorithms we partitioned each hypergraph three times and report average results.

We utilize the column-parallel SpMV implementation [20], which is implemented in C using MPI for interprocess communication. Parallel SpMV times are obtained on a cluster with 19 nodes where each node contains 28 cores (two Intel Xeon E5-2680 v4 CPUs) running at 2.40 GHz clock frequency and 128 GB memory. The nodes are connected by an InfiniBand FDR 56 Gbps network.

4.3 Dataset

We conduct experiments on a very large set of sparse matrices obtained from the SuiteSparse Matrix Collection (formerly known as the University of Florida Sparse Matrix Collection) [21]. We select square (both symmetric and unsymmetric) matrices that have more than 100K and less than 51M rows/columns. The number of nonzeros of these matrices is in the range from 207K to 1.1B. The collection contains 358 such matrices. We exclude those matrices that does not satisfy one of the following two conditions:

i) the row-net hypergraph partitioning in the first phase does not incur empty parts,
ii) the communication hypergraph in the second phase contains more than 100 vertices per part on average.

Condition (i) prevents unrealistic results, whereas condition (ii) ensures partitioning quality in the second phase. The resulting dataset contains 313 matrices for $K = 512$ processors.

As described in Section 2.3.1, the deficiency of the existing reduce-communication hypergraph model increases with increasing irregularity on the net degrees. Therefore, in order to better show the validity of the proposed vertex (re)weighting scheme, we group test matrices according to the coefficient of variation (CV) values on the net degrees of their communication hypergraphs. Here, CV refers to the ratio of the standard deviation to the mean of the net degrees. Recall that the net degree in a communication hypergraph also refers to the number of partial results produced by the respective processor. We use six matrix groups, denoted by $CV_{>0.50}$, $CV_{>0.30}$, $CV_{>0.20}$, $CV_{>0.15}$, $CV_{>0.10}$, and $CV_{>0.00}$ (all matrices). CV_α consists of matrices whose corresponding CV value is greater than α. Note that the set of matrices in a group associated with a smaller value is a superset of matrices in a group associated with a larger value.

Table 1 displays properties of the test matrices as well as properties of their reduce-communication hypergraphs. In the table, the first column shows the lower bound of the CV value of the matrices in each group. The second column shows the number of matrices in the corresponding CV group. In the table, the rest of the columns show the values averaged over the matrices of each CV group. The third and fourth columns show the number of rows/columns and nonzeros, whereas the fifth column shows the number of nonzeros per row/column. The following two columns show maximum number of nonzeros per row and column, respectively.

In Table 1, the last six columns show properties of reduce-communication hypergraphs. The first of those six

TABLE 1
Properties of Test Matrices and Their Reduce-Communication Hypergraphs

CV	sparse matrices						reduce communication hypergraph					
	number of			avg nnz	max nnz per		# of reduce	vtx degree		net degree		
	matrices	rows/cols	nonzeros	per row/col	row	col	tasks	avg	max	min	avg	max
> 0.50	32	615,123	9,335,833	15.18	971	3,219	156,698	3.56	44	105	1,082	7,153
> 0.30	70	651,939	8,855,071	13.58	684	1,453	136,263	3.67	39	115	968	4,445
> 0.20	148	445,686	4,982,116	11.18	280	387	86,634	3.06	19	105	512	1,426
> 0.15	206	499,288	6,641,398	13.30	186	235	109,187	2.97	15	160	627	1,479
> 0.10	302	575,028	7,892,009	13.72	98	114	119,804	2.69	11	202	625	1,248
ALL	313	555,892	7,616,780	13.70	94	109	121,017	2.71	11	212	635	1,248

columns shows the average number of reduce-tasks obtained from column-wise partitioning (using row-net hypergraph model) of the matrices in the respective *CV* group, i.e., number of free vertices in the communication hypergraph. The second and third columns show average and maximum free-vertex degree of those hypergraphs, respectively. Note that all fixed-vertices have a unit degree. The last three columns show the minimum, average, and maximum net degree of those hypergraphs, respectively.

4.4 Results

Performance results are displayed in three tables and two figures. In all tables, the first row shows actual values averaged over each *CV* group, whereas the second row shows the normalized values with respect to respective baseline for each *CV* group.

Table 2 is introduced to show the performance of the proposed SWAP-OUTCAST algorithm in eliminating outcast vertices. The table compares $\mathrm{RC}^{\mathrm{s}}_{\mathrm{vw}}$ against $\mathrm{RC}_{\mathrm{vw}}$ in terms of the ratio of the number of outcast vertices to the total number of free vertices. As seen in the table, the proposed SWAP-OUTCAST algorithm achieves approximately 13 percent less outcast vertices on average. Furthermore, this performance improvement does not change much according to the *CV* group.

Table 3 compares the relative performance of the three different RC models in terms of multiple communication cost metrics as well as parallel runtime on 512 processors.

TABLE 2
Outcast Vertex Elimination for $K = 512$

CV	outcast vertex ratio	
	$\mathrm{RC}_{\mathrm{vw}}$	$\mathrm{RC}^{\mathrm{s}}_{\mathrm{vw}}$
> 0.50	84%	76%
	1.00	0.91
> 0.30	82%	74%
	1.00	0.91
> 0.20	75%	67%
	1.00	0.89
> 0.15	75%	66%
	1.00	0.88
> 0.10	73%	64%
	1.00	0.87
ALL	74%	64%
	1.00	0.87

The communication cost metrics include maximum send volume, average volume, maximum send message, and average message. In the table, after the *CV* column, each one of the 3-column groups of the 12 columns compares the three RC models in terms of one of the above-mentioned communication cost metrics averaged over the respective *CV* group. Here, average volume and average message values refer to the total communication volume and total number of messages divided by the number of processors. We prefer to report average values instead of total values, because average values give a better feeling on how much the maximum values deviate from the average values.

As seen in Table 3, in terms of the maximum send volume metric, both $\mathrm{RC}_{\mathrm{vw}}$ and $\mathrm{RC}^{\mathrm{s}}_{\mathrm{vw}}$ perform significantly better than RC_{b}, where $\mathrm{RC}^{\mathrm{s}}_{\mathrm{vw}}$ is the clear winner. The performance gap between the proposed RC schemes ($\mathrm{RC}_{\mathrm{vw}}$ and $\mathrm{RC}^{\mathrm{s}}_{\mathrm{vw}}$) and the baseline RC_{b} scheme increases with increasing *CV* values. For example, $\mathrm{RC}^{\mathrm{s}}_{\mathrm{vw}}$ achieves a 6 percent improvement over RC_{b} for the matrices in $CV_{>0.10}$ group and this improvement increases to 8, 9, 15, and 19 percent for the matrices in $CV_{>0.15}$, $CV_{>0.20}$, $CV_{>0.30}$, and $CV_{>0.50}$ groups, respectively. This is expected since the irregularity on the net degrees increases with the increasing *CV* value.

In terms of the average/total communication volume metric, $\mathrm{RC}_{\mathrm{vw}}$ performs slightly worse than RC_{b}, whereas $\mathrm{RC}^{\mathrm{s}}_{\mathrm{vw}}$ performs slightly better than RC_{b} and considerably better than $\mathrm{RC}_{\mathrm{vw}}$. This is also expected since neither RC_{b} nor $\mathrm{RC}_{\mathrm{vw}}$ has explicit effort towards decreasing total communication volume due to the outcast vertex assignments, whereas $\mathrm{RC}^{\mathrm{s}}_{\mathrm{vw}}$ tries to decrease the number of such assignments by utilizing the SWAP-OUTCAST algorithm.

As seen in Table 3, the amount of performance improvement of $\mathrm{RC}^{\mathrm{s}}_{\mathrm{vw}}$ over $\mathrm{RC}_{\mathrm{vw}}$ is similar in the maximum send volume and the average volume metrics for all matrices on average. However, for the matrices in the groups with high *CV* values, this performance improvement is much more pronounced in the maximum send volume metric than the average volume metric. For example, for the matrices in the $CV_{>0.50}$ group, the performance gap between $\mathrm{RC}^{\mathrm{s}}_{\mathrm{vw}}$ and $\mathrm{RC}_{\mathrm{vw}}$ is 9 percent in the maximum send volume metric, whereas this improvement is only 3 percent in the average volume metric. This is because, the SWAP-OUTCAST algorithm eliminates much larger number of outcast vertices from the processor/part which produces the largest number of partial results compared to average. For example, for `barrier2-9` matrix with $CV = 0.67$, the performance gap between $\mathrm{RC}^{\mathrm{s}}_{\mathrm{vw}}$ and $\mathrm{RC}_{\mathrm{vw}}$ is 30 percent in the maximum send

TABLE 3
Comparison of Communication Metrics and Parallel Runtimes for $K = 512$

CV	volume of communication						number of messages						parallel runtime		
	maximum			average			maximum			average					
	RC_b	RC_{vw}	RC^s_{vw}	RC_b	RC_{vw}	RC^s_{vw}	RC_b	RC_{vw}	RC^s_{vw}	RC_b	RC_{vw}	RC^s_{vw}	RC_b	RC_{vw}	RC^s_{vw}
> 0.50	7,029	6,266	5,718	1,013	1,019	994	145	56	47	35	16	14	2.28	2.03	1.64
	1.00	0.89	0.81	1.00	1.01	0.98	1.00	0.38	0.32	1.00	0.47	0.41	1.00	0.89	0.72
> 0.30	4,328	3,971	3,697	889	902	880	106	48	41	32	17	15	1.95	1.62	1.36
	1.00	0.92	0.85	1.00	1.01	0.99	1.00	0.46	0.39	1.00	0.51	0.45	1.00	0.83	0.70
> 0.20	1,369	1,301	1,240	446	458	444	48	29	25	17	11	10	0.78	0.69	0.59
	1.00	0.95	0.91	1.00	1.03	0.99	1.00	0.61	0.52	1.00	0.65	0.57	1.00	0.88	0.76
> 0.15	1,419	1,365	1,308	545	561	540	42	27	23	16	11	9	0.71	0.64	0.55
	1.00	0.96	0.92	1.00	1.03	0.99	1.00	0.66	0.55	1.00	0.69	0.59	1.00	0.91	0.77
> 0.10	1,192	1,164	1,123	533	552	528	31	22	19	13	10	8	0.55	0.51	0.44
	1.00	0.98	0.94	1.00	1.04	0.99	1.00	0.73	0.61	1.00	0.76	0.64	1.00	0.93	0.80
ALL	1,192	1,166	1,125	544	563	538	32	23	20	13	10	9	0.58	0.54	0.46
	1.00	0.98	0.94	1.00	1.03	0.99	1.00	0.73	0.62	1.00	0.76	0.65	1.00	0.93	0.80

Volume of communication values are given in terms of the number of the double precision floating point words sent by processors as well as normalized w.r.t. those of RC_b. *Parallel runtimes are given in milliseconds.*

volume metric, whereas this improvement is only 4 percent in the average volume metric. The SWAP-OUTCAST algorithm decreases the number of outcast vertices in the processor that has the maximum send volume by 1,693, whereas it decreases the total number of outcast vertices by 8,784 which refers to an average decrease of 17.16 per processor.

In terms of the maximum and average/total send-message metrics, both RC_{vw} and RC^s_{vw} perform drastically better than RC_b, where RC^s_{vw} is the clear winner. The performance gap between the proposed RC schemes (RC_{vw} and RC^s_{vw}) and the baseline RC_b scheme increases with increasing CV values. For example, in terms of maximum send message metric RC^s_{vw} achieves a 39 percent improvement over RC_b for the matrices in $CV_{>0.10}$ group and this improvement increases to 45, 48, 61, and 68 percent for the matrices in $CV_{>0.15}$, $CV_{>0.20}$, $CV_{>0.30}$, and $CV_{>0.50}$ groups, respectively. For example, in terms of average/total message metric RC^s_{vw} achieves a 36 percent improvement over RC_b for the matrices in $CV_{>0.10}$ group and this improvement increases to 41, 43, 55, and 59 percent for the matrices in $CV_{>0.15}$, $CV_{>0.20}$, $CV_{>0.30}$, and $CV_{>0.50}$ groups, respectively.

The drastic performance improvement of the proposed RC schemes over RC_b scheme may seem to be unexpected since neither RC_{vw} nor RC^s_{vw} directly aims at improving the maximum or average number messages. In the original RC_b scheme, enforcing the balancing constraint of assigning approximately equal number of free vertices among the parts may prevent the HP tool from clustering the pins of especially dense nets to small number of parts. This leads to an unnecessary increase in the connectivity cutsize defined in (6). On the other hand, enforcing the balancing constraint according to the proposed vertex weighting scheme paves the way for the HP tool to cluster the pins of especially dense nets to smaller number of parts. For example, consider a dense net n_k of degree D. In the proposed vertex weighting scheme, n_k connects D free vertices with weight -1, and a fixed vertex v^k_p with weight D. Then the HP tool will have the flexibility of assigning large number of pins of n_k to part $\mathcal{V}_k$ without disturbing the partitioning constraint (7) thus reducing the connectivity of n_k.

It is important to see whether the improvements obtained by the proposed methods in the given communication cost metrics hold in practice. For this purpose, the last three columns of Table 3 show the parallel SpMV times for the three RC models in milliseconds. As seen in the table, both RC_{vw} and RC^s_{vw} achieve significantly better parallel SpMV times than RC_b, where RC^s_{vw} is the fastest. The performance gap between the proposed RC schemes (RC_{vw} and RC^s_{vw}) and the baseline RC_b scheme in general increases with increasing CV values. For example, RC^s_{vw} achieves a 20 percent improvement over RC_b for the matrices in $CV_{>0.10}$ group and this improvement increases to 23, 24, 30, and 28 percent for the matrices in $CV_{>0.15}$, $CV_{>0.20}$, $CV_{>0.30}$, and $CV_{>0.50}$ groups, respectively.

Fig. 6 compares performance of the proposed RC^s_{vw} scheme against the baseline RC_b scheme in terms of speedup curves on six different matrices in the dataset. As seen in the figure, RC^s_{vw} achieves much better scalability then RC_b until 512 processors.

We introduce Fig. 7 to show the variation of the performance of the proposed RC^s_{vw} method against the baseline RC_b method for varying number of processors from $K = 64$ to $K = 1024$. The figure shows the performance variation in terms of all four communication cost metrics for $CV_{>0.50}$ and $CV_{>0.10}$. As seen in the figure, RC^s_{vw} performs better than RC_b in all metrics for all processor counts. For irregular instances ($CV_{>0.50}$), in volume-based-metrics, the performance gap between RC^s_{vw} and RC_b decreases with increasing number of processors until 256, whereas it remains to be same for 512 and 1024 processors. For $CV_{>0.50}$, in latency-based-metrics, the performance gap between RC^s_{vw} and RC_b does not change considerably with varying number of processors. For relatively regular instances ($CV_{>0.10}$), in all metrics, the performance gap between RC^s_{vw} and RC_b does not change considerably. Comparison of $CV_{>0.50}$ and $CV_{>0.10}$ curves show that the performance gap between RC^s_{vw} and RC_b increases with increasing irregularity of SpMV instances.

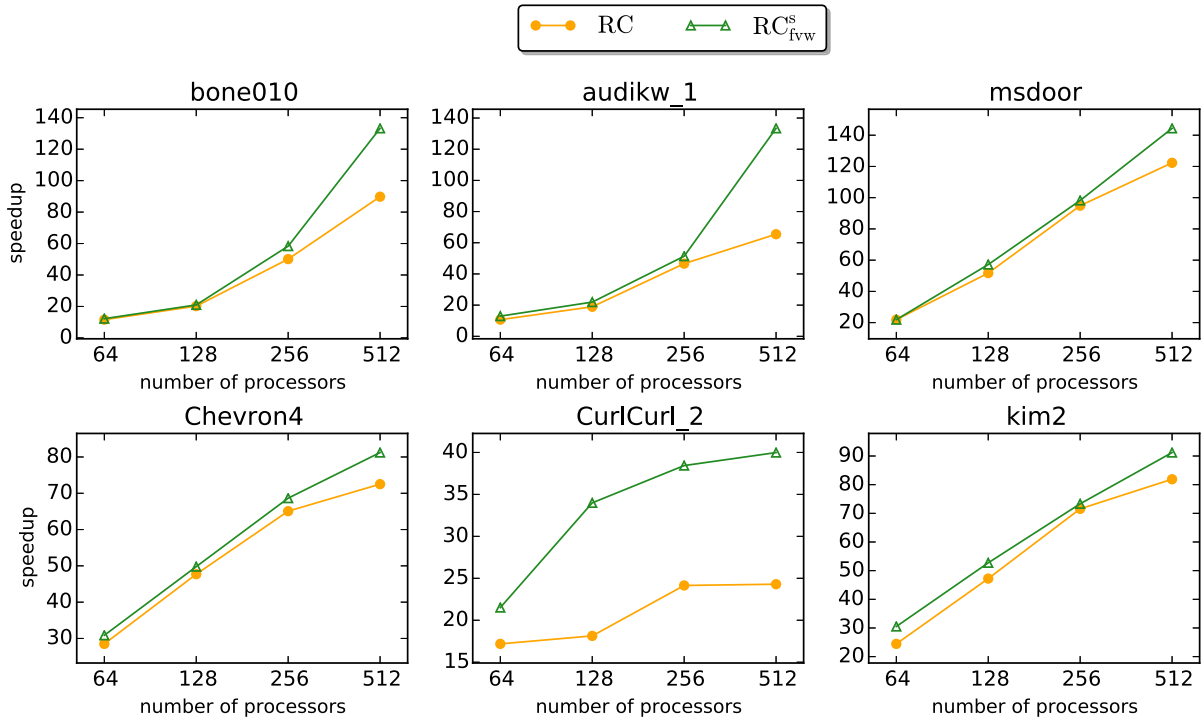

Fig. 6. Strong scaling curves for column-parallel SpMV obtained by $\mathrm{RC_b}$ and $\mathrm{RC^s_{vw}}$.

Table 4 displays sequential partitioning times of the three RC schemes. The table also shows the sequential partitioning times of the row-net hypergraph model. The latter partitioning times are given in order to show additional overhead incurred by the use of the three communication hypergraph models. The proposed RC schemes achieve a significant performance improvement over $\mathrm{RC_b}$ as displayed in the previous tables at the expense of increasing the partitioning overhead of $\mathrm{RC_b}$ by only 2 percent on average. Furthermore, the partitioning time of proposed RC schemes remain below 21 percent of the partitioning time of the row-net hypergraph model on average. In other words, the use of the proposed RC schemes incurs negligible additional overhead compared to the preprocessing overhead introduced by the first phase.

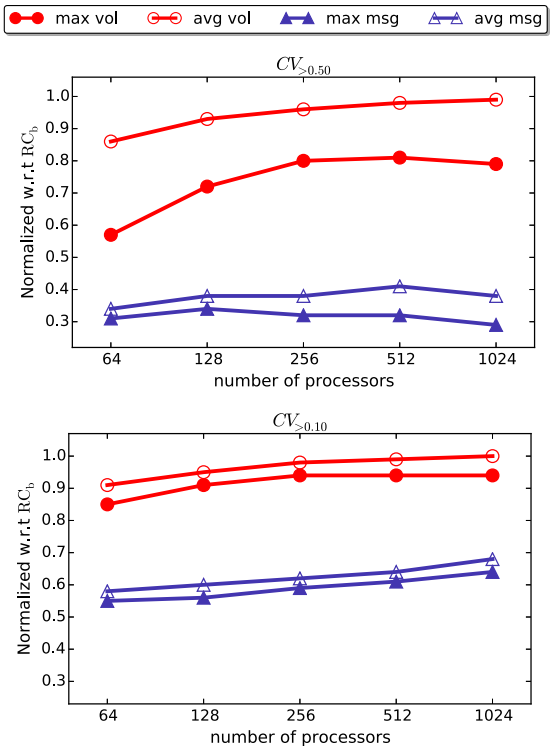

Fig. 7. Variation of communication cost metrics for $\mathrm{RC^s_{vw}}$ with respect to $\mathrm{RC_b}$ for $CV_{>0.50}$ (top) and $CV_{>0.10}$ (bottom).

The amortization analysis for the proposed $\mathrm{RC^s_{vw}}$ method is as follows for $CV_{>0.50}$ on $K = 512$ processors. The average parallel SpMV runtime decreases by 0.64 milliseconds as seen in Table 3. This improvement in parallel runtime is achieved at the expense of the additional partitioning time incurred by $\mathrm{RC^s_{vw}}$. Under 20 percent efficiency assumption (102.4× speedup), the parallel partitioning time will be 57.9 milliseconds as seen in Table 4. So the use of $\mathrm{RC^s_{vw}}$ in the

TABLE 4
Sequential 512-Way Partitioning Times (Seconds)

CV	row-net	$\mathrm{RC_b}$	$\mathrm{RC_{vw}}$	$\mathrm{RC^s_{vw}}$
> 0.50	27.63	5.56	5.41	5.93
	1.00	0.20	0.20	0.21
> 0.30	32.94	4.93	4.95	5.40
	1.00	0.15	0.15	0.16
> 0.20	14.53	2.20	2.30	2.51
	1.00	0.15	0.16	0.17
> 0.15	17.73	3.12	3.23	3.28
	1.00	0.18	0.18	0.19
> 0.10	17.93	3.59	3.77	3.66
	1.00	0.20	0.21	0.20
ALL	17.46	3.64	3.83	3.73
	1.00	0.21	0.22	0.21

second phase amortizes in about 90 repeated parallel SpMV operations with the same coefficient matrix (or coefficient matrices having same sparsity pattern).

5 Conclusion

We focused on the following two deficiencies of the reduce-communication hypergraph model: failing to correctly encapsulate send-volume balancing and increasing the total communication volume due to assigning reduce-tasks to the processors that do not produce partial results for them. For addressing the first deficiency, we proposed a novel vertex weighting scheme so that part weights correctly encode send-volume loads of processors. For addressing the second deficiency, we proposed a swap-based bipartition-refinement method within RB framework for reducing the above-mentioned increase in the communication volume.

We tested the performance of the proposed models on reduce-communication hypergraphs arising in column-parallel SpMV for a wide range of large sparse matrices. Compared to the baseline reduce-communication hypergraph model, the proposed model obtains much better communication-task-to-processor assignments that lead to significantly faster parallel SpMV. The performance gap between the proposed model and the existing model increases with increasing irregularity in the net degrees of the reduce-communication hypergraphs.

As a future work, a two-constraint formulation can be utilized to encode reducing both send and receive volumes separately. For the second constraint, the following vertex weighting scheme will encode reducing maximum receive volume: fixed vertices are assigned unit weights, whereas free vertices are assigned weights equal to their degree. Here, the degree of a vertex refers to the number of nets that connect the respective vertex.

Acknowledgments

Computing resources used in this work were provided by the National Center for High Performance Computing of Turkey (UHeM) under Grant number 4005072018. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

References

[1] Ü. V. Çatalyürek and C. Aykanat, "Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication," *IEEE Trans. Parallel Distrib. Syst.*, vol. 10, no. 7, pp. 673–693, Jul. 1999.

[2] Ü. V. Çatalyürek, C. Aykanat, and B. Uçar, "On two-dimensional sparse matrix partitioning: Models, methods, and a recipe," *SIAM J. Scientific Comput.*, vol. 32, no. 2, pp. 656–683, 2010.

[3] R. H. Bisseling and W. Meesen, "Communication balancing in parallel sparse matrix-vector multiplication," *Electron. Trans. Numerical Anal.*, vol. 21, pp. 47–65, 2005.

[4] B. Vastenhouw and R. H. Bisseling, "A two-dimensional data distribution method for parallel sparse matrix-vector multiplication," *SIAM Rev.*, vol. 47, no. 1, pp. 67–95, 2005.

[5] O. Fortmeier, H. Bücker, B. O. FaggingerAuer, and R. H. Bisseling, "A new metric enabling an exact hypergraph model for the communication volume in distributed-memory parallel applications," *Parallel Comput.*, vol. 39, no. 8, pp. 319–335, 2013.

[6] D. M. Pelt and R. H. Bisseling, "A medium-grain method for fast 2D bipartitioning of sparse matrices," in *Proc. IEEE 28th Int. Parallel Distrib. Process. Symp.*, 2014, pp. 529–539.

[7] B. Uçar and C. Aykanat, "Encapsulating multiple communication-cost metrics in partitioning sparse rectangular matrices for parallel matrix-vector multiplies," *SIAM J. Scientific Comput.*, vol. 25, no. 6, pp. 1837–1859, 2004.

[8] B. Uçar and C. Aykanat, "Minimizing communication cost in fine-grain partitioning of sparse matrices," in *Proc. Int. Symp. Comput. Inf. Sci.*, 2003, pp. 926–933.

[9] O. Selvitopi and C. Aykanat, "Reducing latency cost in 2D sparse matrix partitioning models," *Parallel Comput.*, vol. 57, pp. 1–24, 2016.

[10] K. Akbudak, O. Selvitopi, and C. Aykanat, "Partitioning models for scaling parallel sparse matrix-matrix multiplication," *ACM Trans. Parallel Comput.*, vol. 4, no. 3, pp. 13:1–13:34, Jan. 2018.

[11] O. Selvitopi, S. Acer, and C. Aykanat, "A recursive hypergraph bipartitioning framework for reducing bandwidth and latency costs simultaneously," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 2, pp. 345–358, Feb. 2017.

[12] S. Acer, O. Selvitopi, and C. Aykanat, "Optimizing nonzero-based sparse matrix partitioning models via reducing latency," *J. Parallel Distrib. Comput.*, vol. 122, pp. 145–158, 2018.

[13] S. Acer, O. Selvitopi, and C. Aykanat, "Improving performance of sparse matrix dense matrix multiplication on large-scale parallel systems," *Parallel Comput.*, vol. 59, pp. 71–96, 2016.

[14] Ü. V. Çatalyürek, M. Deveci, K. Kaya, and B. Uçar, "UMPa: A multi-objective, multi-level partitioner for communication minimization," *Graph Partitioning Graph Clustering*, vol. 588, 2013, Art. no. 53.

[15] M. Deveci, K. Kaya, B. Uçar, and Ü. V. Çatalyürek, "Hypergraph partitioning for multiple communication cost metrics: Model and methods," *J. Parallel Distrib. Comput.*, vol. 77, pp. 69–83, 2015.

[16] V. Kumar, A. Grama, A. Gupta, and G. Karypis, *Introduction to Parallel Computing*. San Francisco, CA, USA: Benjamin/Cummings, 1994.

[17] B. Hendrickson and T. Kolda, "Partitioning rectangular and structurally unsymmetric sparse matrices for parallel processing," *SIAM J. Scientific Comput.*, vol. 21, no. 6, pp. 2048–2072, 2000.

[18] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: Applications in VLSI domain," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 7, no. 1, pp. 69–79, Mar. 1999.

[19] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM J. Scientific Comput.*, vol. 20, no. 1, pp. 359–392, 1998.

[20] B. Uçar and C. Aykanat, "A library for parallel sparse matrix-vector multiplies," Bilkent Univ., Ankara, Turkey, Tech. Rep. BU-CE-0506, 2005.

[21] T. A. Davis and Y. Hu, "The University of Florida sparse matrix collection," *ACM Trans. Math. Softw.*, vol. 38, no. 1, pp. 1–25, 2011.

M. Ozan Karsavuran received the BS and MS degrees in computer engineering from Bilkent University, Turkey, in 2012 and 2014, respectively, where he is currently working toward the PhD degree. His research interests include combinatorial scientific computing, graph and hypergraph partitioning for sparse matrix and tensor computations, and parallel computing in distributed and shared memory systems.

Seher Acer received the BS, MS and PhD degrees in computer engineering from Bilkent University, Turkey. She is currently a postdoctoral researcher at Center for Computing Research, Sandia National Laboratories, Albuquerque, New Mexico. Her research interests include parallel computing and combinatorial scientific computing with a focus on partitioning sparse irregular computations.

Cevdet Aykanat received the BS and MS degrees from Middle East Technical University, Turkey, both in electrical engineering, and the PhD degree from Ohio State University, Columbus, in electrical and computer engineering. He worked at the Intel Supercomputer Systems Division, Beaverton, Oregon, as a research associate. Since 1989, he has been affiliated with the Department of Computer Engineering, Bilkent University, Turkey, where he is currently a professor. His research interests include parallel computing and its combinatorial aspects. He is the recipient of the 1995 Investigator Award of The Scientific and Technological Research Council of Turkey and 2007 Parlar Science Award. He has served as an associate editor of the *IEEE Transactions of Parallel and Distributed Systems* between 2009 and 2013.