A General Graph Model For Representing Exact Communication Volume in Parallel Sparse Matrix–Vector Multiplication

Aleksandar Trifunović and William Knottenbelt
{at701,wjk}@doc.ic.ac.uk

Department of Computing, Imperial College London, South Kensington Campus, London SW7 2AZ, UK

Abstract. In this paper, we present a new graph model of sparse matrix decomposition for parallel sparse matrix–vector multiplication. Our model differs from previous graph-based approaches in two main respects. Firstly, our model is based on edge colouring rather than vertex partitioning. Secondly, our model is able to correctly quantify and minimise the total communication volume of the parallel sparse matrix–vector multiplication while maintaining the computational load balance across the processors. We show that our graph edge colouring model is equivalent to the fine-grained hypergraph partitioning-based sparse matrix decomposition model. We conjecture that the existence of such a graph model should lead to faster serial and parallel sparse matrix decomposition heuristics and associated tools.

1 Introduction

Parallel sparse matrix–vector multiplication is the core operation in iterative solvers for large-scale linear systems and eigensystems. Major application areas include Markov modelling, linear programming and PageRank computation.

Efficient parallel sparse matrix–vector multiplication requires intelligent *a priori* partitioning of the sparse matrix non-zeros across the processors to ensure that interprocessor communication is minimised subject to a load balancing constraint. The problem of sparse matrix decomposition can be reformulated in terms of a graph or hypergraph partitioning problem. These partitioning problems are NP-hard [10], so (sub-optimal) heuristic algorithms are used in practice. The resulting graph or hypergraph partition is then used to direct the distribution of matrix elements across processors.

The limits of the existing graph partitioning approaches are outlined in [11, 8, 4]. For example, in the case of one-dimensional row-wise or column-wise partitioning of a sparse matrix for parallel sparse matrix–vector multiplication, existing graph models cannot optimise the exact communication volume; instead, they operate indirectly by optimising an upper bound on the communication volume.

On the other hand, hypergraph models that correctly represent the total communication volume have been proposed and are thus preferred to graph models in practical applications. Moreover, recently two parallel hypergraph partitioning algorithms have also been developed and implemented [17, 16, 15, 7]. However, graph models do have the advantage that heuristic algorithms operating on graphs are faster and are significantly easier to parallelise than heuristic algorithms that operate on hypergraphs [15, 7].

This paper presents a bipartite graph model for parallel sparse matrix–vector multiplication that correctly models the total interprocessor communication volume while maintaining the computational load balance. The graph model is derived from the fine-grained hypergraph model presented by Çatalyürek and Aykanat in [5]. The edges in the graph model the non-zeros in the matrix and thus instead of partitioning the set of vertices, as in existing graph and hypergraph sparse matrix decomposition models, our model requires the colouring of the edges of the graph so that a colouring objective is minimised. Whereas the widely accepted meaning of the phrase ”edge colouring” is that the edges of the graph are coloured such that edges incident on the same vertex have different colours, the edge colouring that we seek imposes no such restriction, i.e. we admit colourings where distinct edges incident on the same vertex are of the same colour. The colouring objective correctly models the total interprocessor communication volume, while the computational load balance is maintained by the constraint limiting the number of edges (matrix non-zeros) that can be assigned the same colour.

We anticipate that the advantages of our graph model over existing hypergraph models will be twofold. Firstly, heuristic algorithms for minimising the edge-colouring objective in a graph should be faster than heuristic algorithms for the corresponding hypergraph partitioning problem and secondly, the edge-colouring algorithms should yield more efficient parallel algorithms than their hypergraph partitioning counterparts, as indicated by respective state-of-the-art algorithms for graph and hypergraph partitioning.

The remainder of this paper is organized as follows. Section 2 describes the models used in sparse matrix decomposition for parallel sparse matrix–vector multiplication. Section 3 describes the main contribution of our paper, the graph edge colouring model. Section 4 concludes and considers future directions for this work.

2 Decomposition Models for Parallel Sparse Matrix–Vector Multiplication

2.1 Preliminaries

Consider a sparse $m \times n$ matrix $\mathbf{A}$. We require that the sparse matrix–vector product $\mathbf{Ax} = \mathbf{b}$ is distributed across p processors, where $\mathbf{x}$ and $\mathbf{b}$ are dense n- and m-vectors respectively. In [19], Vastenhouw and Bisseling note that the natural parallel algorithm, with an arbitrary non-overlapping distribution of the matrix and the vectors across the processors, has the following general form:

1. Each processor sends its components x_j to those processors that possess a non-zero a_{ij} in column j.

2. Each processor computes the products $a_{ij}x_j$ for its non-zeros a_{ij} and adds the results for the same row index i. This yields a set of contributions b_{is}, where s is the processor identifier $1 \leq s \leq p$.
3. Each processor sends its non-zero contributions b_{is} to the processor that is assigned vector element b_i.
4. Each processor adds the contributions received for its components b_i, giving $b_i = \sum_{s=1}^{p} b_{is}$.

In common with other authors (e.g. [19, 2]), we assume that the processors synchronize globally between the above phases. The computational requirement of step 2 dominates that of step 4 [19]; henceforth we assume that the computational load of the entire parallel sparse matrix–vector multiplication algorithm can be represented by the computational load induced during step 2 only.

It is noted in [2] that the decomposition of the sparse matrix $\mathbf{A}$ to the p processors may be done in one of the following ways:

1. One-dimensional [4]; entire rows (or columns) of the matrix are allocated to individual processors. This has the effect of making the communication step 3 (or 1 in the column case) in the parallel sparse matrix–vector multiplication pipeline redundant.
2. Two-dimensional Cartesian [6]; each processor receives a submatrix defined by a partition of rows and columns of $\mathbf{A}$.
3. Two-dimensional non-Cartesian with the Mondriaan structure [19]; obtained by recursively bipartitioning the matrix in either the row or column direction.
4. Arbitrary (fine-grained) two-dimensional [5]; each non-zero is assigned individually to a processor. This is the most general decomposition.

The above decompositions of the sparse matrix $\mathbf{A}$ to the p processors are usually modelled as a graph or hypergraph partitioning problem.

2.2 Graph and Hypergraph Partitioning

Given a finite set of m vertices, $V = \{v_1, \ldots, v_m\}$, a hypergraph on V is a set system, here denoted $H(V, \mathcal{E})$, such that $\mathcal{E} \subset \mathcal{P}(V)$, where $\mathcal{P}(V)$ is the power set of V. The set $\mathcal{E} = \{e_1, \ldots, e_n\}$ is said to be the set of hyperedges of the hypergraph. When $\mathcal{E} \subset V^{(2)}$, each hyperedge has cardinality two and the resulting set system is known as a *graph*. Henceforth, definitions are given in terms of hypergraphs (although they also hold for graphs) and whenever we specifically need to distinguish between a graph and a hypergraph, the graph shall be denoted by $G(V, \mathcal{E})$.

A hyperedge $e \in \mathcal{E}$ is said to be incident on a vertex $v \in V$ in a hypergraph $H(V, \mathcal{E})$ if, and only if, $v \in e$. The incidence matrix of a hypergraph $H(V, \mathcal{E})$, $V = \{v_1, \ldots, v_m\}$ and $\mathcal{E} = \{e_1, \ldots, e_n\}$, is the $m \times n$ matrix $\mathbf{M} = (m_{ij})$, with entries

$$m_{ij} = \begin{cases} 1 \text{ if } v_i \in e_j \\ 0 \text{ otherwise} \end{cases} \tag{1}$$

In a hyperedge- and vertex-weighted hypergraph, each hyperedge $e \in \mathcal{E}$ and each vertex $v \in V$ are assigned a scalar weight.

A partition Π of a hypergraph $H(V, \mathcal{E})$ is a finite collection of subsets of V (called parts), such that $P \cap P' = \emptyset$ is true for all $P, P' \in \Pi$ and $\bigcup_i P_i = V$. The weight $w(P)$ of a part $P \in \Pi$ is given by the sum of the constituent vertex weights. Given a real-valued balance criterion $0 < \epsilon < 1$, the k-way hypergraph partitioning problem requires a k-way partition Π that satisfies

$$w(P_i) < (1 + \epsilon)W_{avg} \tag{2}$$

for all $1 \leq i \leq k$ (where $W_{avg} = \sum_{i=1}^{k} w(P_i)/k$) and is such that some partitioning objective function f_p is minimised.

For sparse matrix decomposition problems, the partitioning objective of interest is the $k-1$ metric [4]. Here, each hyperedge $e \in \mathcal{E}$ contributes $(\lambda(e) - 1)w(e)$ to the objective function, where $\lambda(e)$ is the number of parts spanned by, and $w(e)$ the weight of, hyperedge e:

$$f_p(\Pi) = \sum_{e \in \mathcal{E}} (\lambda(e) - 1)w(e) \tag{3}$$

Note that for graph partitioning this reduces to the edge-cut metric, since the cardinality of each edge is two.

2.3 Related Work

In [12], Hendrickson and Kolda outline a bipartite graph partitioning-based model for decomposition of a general rectangular non-symmetric sparse matrix. The non-zero structure of a sparse matrix $\mathbf{A}$ corresponds to an undirected bipartite graph $G(V, \mathcal{E})$. We have $V = \mathcal{R} \cup \mathcal{C}$, such that $\mathcal{R} = \{r_1, \ldots, r_m\}$ and $\mathcal{C} = \{c_1, \ldots, c_n\}$ and $(r_i, c_j) \in \mathcal{E}$ if and only if $a_{ij} \neq 0$. In the row decomposition-based model, the weight of the row vertices $\{v \in \mathcal{R}\}$ is given by the number of non-zeros in each row. The column vertices $\{v \in \mathcal{C}\}$ and the edges have unit weights. The partitioning constraint requires that the total weight of vertices from $\mathcal{R}$ allocated to each processor is approximately the same. It is noted that an exact representation of the communication volume may be given by $\sum_i (\lambda(c_i) - 1)$, where $\lambda(c_i)$ is the number of distinct parts that neighbours of $c_i \in \mathcal{C}$ have been allocated to. The authors chose to approximate this metric with the number of edges cut (and thus approximately model the total communication volume) because of the difficulties in minimising the metric that yields the exact communication volume.

Incidentally, in this work we derive a bipartite graph with the same topological structure; however, we use a different weighting on the vertices so as to model the most general sparse matrix decomposition; further, we correctly quantify the total communication volume using a graph edge colouring metric.

Hypergraph partitioning was first applied in sparse matrix decomposition for parallel sparse matrix–vector multiplication by Çatalyürek and Aykanat in [4]. They proposed a one-dimensional decomposition model for a square non-symmetric sparse matrix; without loss of generality, we here describe the row-based decomposition.

The sparsity pattern of the sparse matrix $\mathbf{A}$ is interpreted as the incidence matrix of a hypergraph $H(V, \mathcal{E})$. The rows of $\mathbf{A}$ are interpreted as the vertices and the columns of $\mathbf{A}$ the hyperedges in H. The weight of vertex $v_i \in V$ (modelling row i in $\mathbf{A}$) is given by the number of non-zero elements in row i. The vector elements x_i and b_i are allocated to the processor that is allocated row i of $\mathbf{A}$. Because the authors assumed a symmetric partitioning of the vectors $\mathbf{x}$ and $\mathbf{b}$ (entries x_i and b_i always allocated to the same processor), in order for the $k-1$ metric to correctly represent the total communication volume of the parallel sparse matrix–vector multiplication, a "dummy" non-zero a_{ii} is added to the model whenever $a_{ii} = 0$. Note that in general the addition of dummy non-zeros is not necessary. For a general $m \times n$ sparse matrix, provided that the vector component x_i is assigned to a processor allocated a non-zero in column i and the vector component b_j is assigned to a processor allocated a non-zero in row j, the $k-1$ metric will correctly represent the total communication volume of the parallel sparse matrix–vector multiplication under a one-dimensional decomposition.

In [6], Çatalyürek and Aykanat extend the one-dimensional model to a coarse-grained two-dimensional one. The model yields a cartesian partitioning of the matrix; the rows are partitioned into α sets R_π using the one-dimensional row-based hypergraph partitioning model and the columns are partitioned into β sets C_π using the column-based one-dimensional hypergraph partitioning model. The $p = \alpha\beta$ cartesian products $R_\pi \times C_\pi$ are assigned to processors with the (symmetric) vector distribution given by the distribution of the matrix diagonal.

Vastenhouw and Bisseling [19] propose recursive bipartitioning of the general sparse matrix, alternating between the row and column directions. They show that when partitioning a general sparse matrix, its submatrices can be partitioned independently while still correctly modelling the total communication volume.

In [5], Çatalyürek and Aykanat propose a hypergraph model for the most general sparse matrix decomposition. In this fine-grained model, each $a_{ij} \neq 0$ is modelled by a vertex $v \in V$, so that a p-way partition Π of the hypergraph $H(V, \mathcal{E})$ will correspond to an assignment of the matrix non-zeros to p processors. The causes of communication between processors in steps 1 and 3 of the parallel sparse matrix–vector multiplication pipeline define the hyperedges of the hypergraph model. In step 1, the processor with non-zero a_{ij} requires vector element x_j for computation during step 2. This results in a communication of x_j to the processor assigned a_{ij} if x_j had been assigned to a different processor. The dependence between the non-zeros in column j of the matrix $\mathbf{A}$ and vector element x_j can be modelled by a hyperedge, whose constituent vertices are the non-zeros of column j of the matrix $\mathbf{A}$. Such hyperedges are henceforth called column hyperedges. In [5], a "dummy" non-zero a_{ii} is added to the model if $a_{ii} = 0$, because symmetric partitioning of the vectors $\mathbf{x}$ and $\mathbf{b}$ is used. The construction of hyperedges modelling the communication requirements in step 3 is identical, except that now the communication requirement depends on the partitioning of the vector $\mathbf{b}$. These hyperedges are called row hyperedges. The $k-1$ metric evaluated on partitions of this hypergraph then correctly models the total communication volume.

This fine-grained two-dimensional model is essentially a generalisation of the one dimensional model (in which either only row or only column hyperedges are present). As previously noted in the case of the one-dimensional decomposition, the addition of dummy non-zeros is also not necessary in the two-dimensional model provided the vector component x_j is assigned to a processor allocated a non-zero in column j and the vector component b_i is assigned to a processor allocated a non-zero in row i.

In [18, 2], the problem of vector partitioning is considered with the aim of improving the communication balance and reducing the number of messages sent between the processors, while maintaining the overall communication volume. Provided we are prepared to accept an arbitrary partition of vectors $\mathbf{x}$ and $\mathbf{b}$ across the processors, the problem of vector partitioning is orthogonal to the problem of partitioning the sparse matrix; we know that for any given matrix partition produced by one of the hypergraph partitioning-based decomposition models and the total communication volume given by the $k-1$ metric, there exists a partition of the vectors $\mathbf{x}$ and $\mathbf{b}$ across the processors that yields the claimed total communication volume. Here we will not explicitly consider the problem of vector partitioning when describing our graph-based model, but note instead that incorporating vector partitioning heuristics within a model for the total communication volume is an area of ongoing research [1].

3 The General Graph Model

In this section, we derive our graph colouring-based model for sparse matrix decomposition. We show that our model yields a bipartite graph, which is in fact the same bipartite graph that results from applying the model of Hendrickson and Kolda [12], described in Section 2.3.

3.1 Graph Model Construction

We note that the transpose of the incidence matrix, $\mathbf{M}^T$, defines the *dual* hypergraph $H^*(V^*, \mathcal{E}^*)$ [3]. Duality is commutative, so that $H(V, \mathcal{E})$ is also the dual of $H^*(V^*, \mathcal{E}^*)$.

The hyperedges in H correspond to vertices in H^* and the vertices in H to the hyperedges in H^*. To see this, let $e_i \in \mathcal{E}$ and $e_j \in \mathcal{E}$ denote the i^{th} and j^{th} hyperedges in H. Then, the i^{th} and j^{th} vertices in the dual hypergraph H^* are connected by a hyperedge $e^* \in \mathcal{E}^*$ only if there exists some vertex $v \in V$ such that $v \in e_i$ and $v \in e_j$. The hyperedge e^* in the dual H^* is in direct correspondence with the vertex v. Let $\delta : V \to \mathcal{E}^*$ denote the (invertible) function that maps the vertices in H to hyperedges in H^*.

Let H denote the hypergraph arising from the most general (fine-grained) sparse matrix decomposition model described in Section 2.3 and consider H^*, its dual. Because each vertex v in V (that represents a non-zero in $\mathbf{A}$) is incident on exactly one row and one column hyperedge, the incidence matrix $\mathbf{M}$ has two non-zeros in each row and thus the incidence matrix of the dual $\mathbf{M}^T$ has exactly two non-zeros in each column. Clearly, the dual hypergraph H^* is then a graph. Moreover, this graph is bipartite. To see this, consider vertices $v^*, w^* \in V^*$ that correspond to row hyperedges in the hypergraph arising from the hypergraph model H. Because the row hyperedges share no common vertices in H, v^* and w^* are not connected. A similar argument using column hyperedges shows that (hyper)edges in H^* only connect vertices corresponding to column hyperedges in H with vertices corresponding to row hyperedges in H. Henceforth, when referring to the dual of the hypergraph arising from the two-dimensional fine-grained model, we will denote it $G(V^*, \mathcal{E}^*)$.

To see that a bipartite graph with the same structure as $G(V^*, \mathcal{E}^*)$ also arises from the model of Hendrickson and Kolda [12], described in Section 2.3, note that the sets of vertices $\mathcal{R}$ and $\mathcal{C}$ in Hendrickson and Kolda's model are the sets of vertices in the dual graph corresponding to row and column hyperedges in the fine-grained hypergraph model. Hendrickson and Kolda partition the set $\mathcal{R}$ across the processors, which corresponds to allocating row hyperedges in the fine-grained hypergraph model to processors. Because row hyperedges in the fine-grained model just define the rows of the sparse matrix $\mathbf{A}$, we can (correctly) deduce that the model of Hendrickson and Kolda is in fact a row-wise one-dimensional sparse matrix decomposition.

3.2 The Communication Cost Metric

Having constructed the bipartite graph model $G(V^*, \mathcal{E}^*)$, we define a communication cost metric that correctly models both the total communication volume and the computational load induced on each processor during the parallel sparse matrix–vector multiplication. We know that the hypergraph partitioning problem with the $k-1$ objective achieves this for the hypergraph model $H(V, \mathcal{E})$ with incidence matrix $\mathbf{M}$.

The weight of each vertex $v^* \in V^*$ is set to the weight of the corresponding hyperedge in H and the weight of each (hyper)edge $e^* \in \mathcal{E}^*$ is set to the weight of the corresponding vertex in V. We seek an edge colouring of G (i.e. an assignment of colours to edges in $\mathcal{E}^*$) using p colours and allowing edges incident on the same vertex to be assigned the same colour. Let $c(e^*)$ denote the colour assigned to edge e^* and let C_i, $1 \leq i \leq p$ be the colours used. The colouring must satisfy the following constraint (for each $1 \leq i \leq p$):

$$\sum_{\{e^* \in \mathcal{E}^* | c(e^*) = C_i\}} w(e^*) < (1 + \epsilon) W_{avg} \tag{4}$$

where ϵ is the prescribed balance criterion and $W_{avg} = \sum_{e^* \in \mathcal{E}^*} w(e^*)/p$. Because the weight of each edge $e^* \in \mathcal{E}^*$ is set to the weight of the corresponding vertex in V, the total edge-weight allocated a distinct colour exactly corresponds to the computational load induced on the processor represented by that colour (as there are p colours and p processors).

The colouring objective is defined in terms of the number of distinct colours that are "induced" onto each vertex by edges connected to it. An edge $e^* \in \mathcal{E}^*$, assigned the colour C_i, $1 \leq i \leq p$, also induces the colour C_i onto a vertex $v^* \in V^*$ if and only if $v^* \in e^*$. The colouring objective is to minimise the function f_c given by

$$f_c(G) = \sum_{v^* \in V^*} w(v^*)(\sigma(v^*) - 1) \tag{5}$$

where $\sigma(v^*)$ represents the number of distinct colours induced onto vertex v^* by edges incident on it.

Proposition 1. *Minimising the colouring objective in Equation 5 subject to the constraint in Equation 2 is equivalent to minimising the total communication volume of parallel sparse matrix–vector multiplication while maintaining an overall computational load balance across the processors.*

Proof. Since the colouring constraint in Equation 4 is equivalent to maintaining the imbalance in computational load within a (prescribed) factor of ϵ, it remains to show that Equation 5 yields the total communication volume of parallel sparse matrix–vector multiplication. To achieve this, it is enough to show that the colouring objective $f_c(G)$ for the graph G is equivalent to the $k-1$ hypergraph partitioning objective for (its dual) hypergraph H.

Let $\mathbf{M}$ be the incidence matrix of the hypergraph H, so that $\mathbf{M}^T$ is the incidence matrix of its dual graph G.

Now, consider the computation of the $k-1$ objective ($f_p(\Pi)$ in Equation 3) on a partition Π of H. In order to compute the sum in Equation 3, we need to traverse the matrix $\mathbf{M}$ column-by-column. To compute $\lambda(e_i)$ for column (hyperedge) e_i, we consider each non-zero in the i^{th} column. The allocation of vertices (rows of $\mathbf{M}$) to parts associates a part (processor) with each non-zero in $\mathbf{M}$ so that $\lambda(e_i)$ is given by the number of distinct parts associated with the non-zeros in column i.

To compute $f_c(G)$, we traverse the matrix $\mathbf{M}^T$ row-by-row. The quantity $\sigma(v_i^*)$ is computed by considering each non-zero in the i^{th} row of $\mathbf{M}^T$, so that $\sigma(v_i^*)$ is given by the number of distinct colours associated with the non-zeros in row i. But this is equivalent to traversing the matrix $\mathbf{M}$ column-by-column and computing the number of distinct colours associated with the non-zeros in column i. To obtain the edge colouring of the graph G that is equivalent to the partition Π (in terms of the total communication volume and computational load across the processors), assign colour C_i to edge e^* whenever the vertex $v = \delta^{-1}(e^*)$ is assigned to part P_i, for all colours C_i and parts P_i, $1 \leq i \leq p$. $\square$

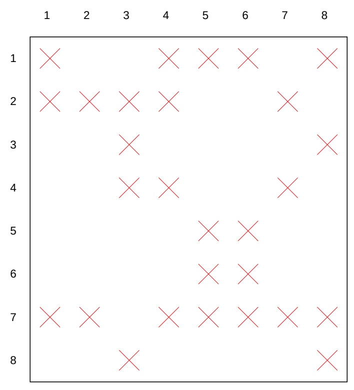

Fig. 1. Sample sparse matrix for decomposition across 2 processors

3.3 Worked Example

Consider the sparse matrix shown in Figure 1, and consider a decomposition of this matrix over two processors. A one-dimensional row-wise partitioning based on a 1D hypergraph model with a 10% balance constraint results in the decomposition shown in Figure 2. Matrix and vector elements assigned to processor 1 are coloured red; those assigned to processor 2 are coloured green. The resulting total communication volume per sparse matrix–vector product is 6, and there is a slight computational load imbalance (15 vs. 13 non-zero matrix elements).

Figure 4 shows our bipartite graph model, with edges coloured so as to minimize the colouring objective of Equation 5 under ideal load balance. The resulting fine-grained sparse matrix decomposition is shown in Figure 3. The resulting total communication volume per sparse matrix–vector product is 3, with no load imbalance. Precisely the same decomposition emerges from a 2D fine-grained hypergraph model based on vertex partitioning.

3.4 A Discussion of the Graph Model

In Section 3.2, we showed that our graph colouring-based model is as expressive as the general fine-grained two-dimensional hypergraph partitioning-based sparse matrix decomposition model. In this section, we briefly discuss possible modifications of existing graph and hypergraph partitioning algorithms to the given graph edge colouring problem and the anticipated advantages over existing hypergraph-based models. Of course, this does not preclude alternative approaches to the edge colouring problem.

The multilevel approach yields algorithms of choice for both graph and hypergraph partitioning [15]. Reduction in complexity of the original problem instance for our graph colouring objective is also possible via the multilevel method.

Consider the coarsening procedure for a graph $G(V^*, \mathcal{E}^*)$. Given vertices $u, v, w \in V^*$ and edges $(u, v), (u, w), (v, w) \in \mathcal{E}^*$, one could match the vertices

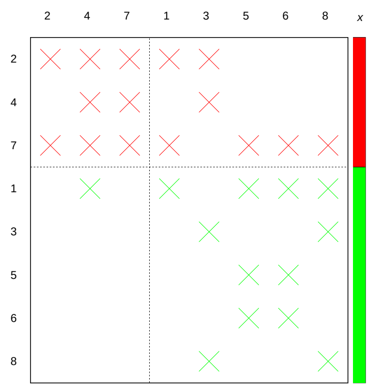

Fig. 2. Row-wise sparse matrix decomposition under 1D hypergraph model

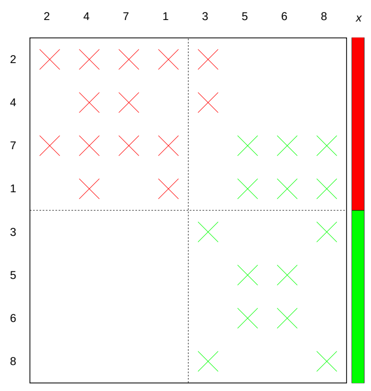

Fig. 3. Fine-grained sparse matrix decomposition under bipartite graph model

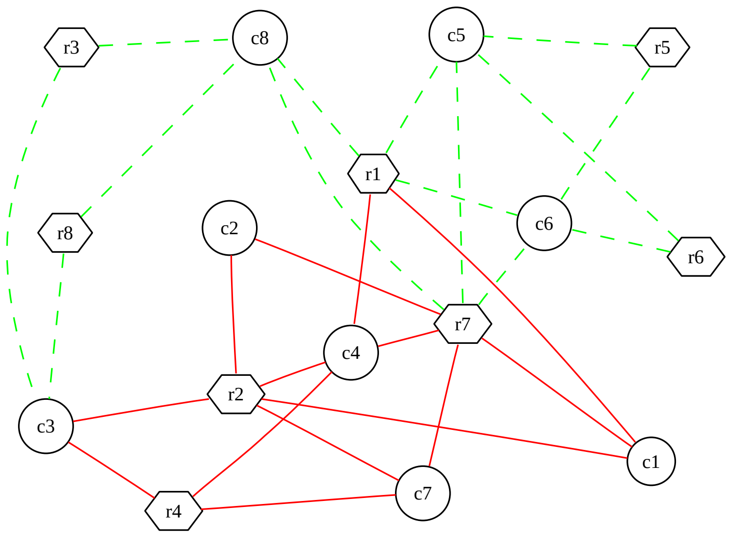

Fig. 4. Bipartite graph model with optimal balanced edge colouring

u and v so that they form a cluster $c = \{u, v\}$ (a vertex in the coarse graph replacing u and v, whose weight is set to the sum of the weights of u and v). Then, the edges (u, w) and (v, w) will be replaced in the coarser graph by the edge (c, w), whose weight is set to be the sum of the weights of the edges (u, w) and (v, w). The edge (u, v) is replaced with the "edge" consisting of only vertex c, (c). However, unlike coarsening in the context of partitioning, we cannot discard the edge (c), since the colour assigned to this edge will (potentially) contribute to the colouring objective. We can, though, replace all duplicate edges with a single instance of the edge, setting its weight as the sum of the weights of the duplicate edges. The only difference between the coarse graphs produced by coarsening algorithms in the context of graph partitioning and the graph colouring, respectively, will be the (at most) $|V^*|$ additional singleton edges in the graph colouring case, where V^* is the set of vertices in the coarsest graph. Like uncoarsening in a multilevel partitioning algorithm, a colouring of the coarse graph can be projected onto the finer graph while maintaining the value of the colouring objective. We thus foresee that coarsening algorithms developed for graph partitioning could be applied in the graph colouring context. We also expect that iterative improvement algorithms such as the Fiduccia and Mattheyses algorithm [9] may be applied to our graph colouring problem. Feasible moves are defined as colour changes of edges, such that the overall colouring imbalance satisfies the prescribed constraints (cf. Equation 2).

We foresee two main advantages of the multilevel graph edge colouring algorithms over their multilevel hypergraph partitioning counterparts in the context of sparse matrix decomposition. As graph partitioning algorithms are in general faster than corresponding hypergraph partitioning algorithms, we conjecture than the graph colouring model will yield faster algorithms than the corresponding hypergraph partitioning algorithms. We also anticipate that the graph colouring algorithms will offer significantly more scope for parallelism than the corresponding hypergraph partitioning algorithms. This is attributed to the fact that the cardinalities of hyperedges in a hypergraph may vary from one to an upper bound given by the cardinality of the vertex set, whereas the cardinality of every edge in a graph is two. Indeed, existing parallel multilevel graph partitioning algorithms have demonstrated more natural parallelism than has been hitherto shown for hypergraph partitioning [13, 14, 20, 7, 15].

4 Conclusion and Future Work

In this paper we have presented a new graph edge colouring-based model of sparse matrix decomposition for parallel sparse matrix–vector multiplication. Unlike previous graph-based models, our model correctly quantifies the total communication volume of parallel sparse matrix–vector multiplication. Because the edge-colouring algorithms operate on a graph instead of a hypergraph (as the hitherto used partitioning algorithms do), we conjecture that our model will lead to faster sparse matrix decomposition algorithms.

The initial focus of our future work will be an implementation of a multilevel graph edge colouring algorithm in order to empirically evaluate the feasibility of our model and provide a benchmark comparison with respect to existing hypergraph partitioning-based models.

References

1. R.H. Bisseling, January 2006. Personal communication.

2. R.H. Bisseling and W. Meesen. Communication balancing in parallel sparse matrix–vector multiplication. *Electronic Transactions on Numerical Analysis: Special Volume on Combinatorial Scientific Computing*, 21:47–65, 2005.
3. B. Bollobás. *Combinatorics.* Cambridge University Press, 1986.
4. U.V. Çatalyürek and C. Aykanat. Hypergraph Partitioning-based Decomposition for Parallel Sparse–Matrix Vector Multiplication. *IEEE Transactions on Parallel and Distributed Systems*, 10(7):673–693, 1999.
5. U.V. Çatalyürek and C. Aykanat. A fine-grain hypergraph model for 2D decomposition of sparse matrices. In *Proc. 8th International Workshop on Solving Irregularly Structured Problems in Parallel*, San Francisco, USA, April 2001.
6. U.V. Çatalyürek and C. Aykanat. *PaToH: Partitioning Tool for Hypergraphs, Version 3.0*, 2001.
7. K.D. Devine, E.G. Boman, R.T. Heaphy, R.H. Bisseling, and U.V. Çatalyürek. Parallel hypergraph partitioning for scientific computing. In *Proc. 20th IEEE International Parallel and Distributed Processing Symposium*, 2006.
8. K.D. Devine, E.G. Boman, R.T. Heaphy, B.A. Hendrickson, J.D. Teresco, J.Faik, J.E. Flaherty, and L.G. Gervasio. New Challenges in Dynamic Load Balancing. *Applied Numerical Mathematics*, 52(2–3):133–152, 2005.
9. C.M. Fiduccia and R.M. Mattheyses. A Linear Time Heuristic For Improving Network Partitions. In *Proc. 19th IEEE Design Automation Conference*, pages 175–181, 1982.
10. M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Co., 1979.
11. B.A. Hendrickson. Graph Partitioning and Parallel Solvers: Has the Emperor No Clothes. In *Proc. Irregular'98*, volume 1457 of *LNCS*, pages 218–225. Springer, 1998.
12. B.A. Hendrickson and T.G. Kolda. Partitioning Rectangular and Structurally Nonsymmetric Sparse Matrices for Parallel Processing. *SIAM Journal of Scientific Computing*, 21(6):248–272, 2000.
13. G. Karypis and V. Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1999.
14. G. Karypis, K. Schloegel, and V. Kumar. *ParMeTiS: Parallel Graph Partitioning and Sparse Matrix Ordering Library, Version 3.0.* University of Minnesota, 2002.
15. A. Trifunović. *Parallel Algorithms for Hypergraph Partitioning.* PhD thesis, Imperial College London, February 2006.
16. A. Trifunović and W.J. Knottenbelt. A Parallel Algorithm for Multilevel k-way Hypergraph Partitioning. In *Proc. 3rd International Symposium on Parallel and Distributed Computing*, pages 114–121, University College Cork, Ireland, July 2004.
17. A. Trifunović and W.J. Knottenbelt. Towards a Parallel Disk-Based Algorithm for Multilevel k-way Hypergraph Partitioning. In *Proc. 5th Workshop on Parallel and Distributed Scientific and Engineering Computing*, April 2004.
18. B. Uçar and C. Aykanat. Encapsulating Multiple Communication-Cost Metrics in Partitioning Sparse Rectangular Matrices for Parallel Matrix–Vector Multiples. *SIAM Journal on Scientific Computing*, 25(6):1837–1859, 2004.
19. B. Vastenhouw and R.H. Bisseling. A Two-Dimensional Data Distribution Method for Parallel Sparse Matrix–Vector Multiplication. *SIAM Review*, 47(1):67–95, 2005.
20. C. Walshaw, M. Cross, and M. G. Everett. Parallel dynamic graph partitioning for adaptive unstructured meshes. *J. Parallel Distrib. Comput.*, 47(2):102–108, 1997.