Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

An exact algorithm for sparse matrix bipartitioning

Daniël M. Pelt[a,*], Rob H. Bisseling[b]

[a] Scientific Computing Group, Centrum Wiskunde & Informatica, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
[b] Mathematical Institute, Utrecht University, P.O. Box 80010, 3508 TA Utrecht, The Netherlands

HIGHLIGHTS

- We present an exact branch-and-bound algorithm for sparse matrix bipartitioning.
- Rows and columns are partitioned instead of nonzeros to reduce computation time.
- For 85% of a test set of small matrices, an optimal bipartitioning was found.
- The largest matrix that was optimally bipartitioned has 129,042 nonzeros.
- A benchmark collection of optimal results will be made publicly available.

ARTICLE INFO

Article history:
Received 16 December 2014
Received in revised form
1 May 2015
Accepted 16 June 2015
Available online 22 June 2015

Keywords:
Branch-and-bound
Exact algorithm
Hypergraph
Parallel computing
Partitioning
Sparse matrix
Sparse matrix–vector multiplication

ABSTRACT

The sparse matrix partitioning problem arises when minimizing communication in parallel sparse matrix–vector multiplications. Since the problem is NP-hard, heuristics are usually employed to find solutions. Here, we present a purely combinatorial branch-and-bound method for computing optimal bipartitionings of sparse matrices, in the sense that they have the lowest communication volume out of all possible bipartitionings obeying a certain load balance constraint. The method is based on a way of partitioning similar to the recently proposed medium-grain heuristic, which reduces the number of solutions to be considered in the branch-and-bound method.

We applied the proposed optimal bipartitioner to find the optimal communication volume of all matrices of the University of Florida sparse matrix collection with 1000 nonzeros or less. For 85% of the matrices, an optimal bipartitioning was found within a single day of computation and for 58% even within a second. We also present optimal results for selected larger matrices, up to 129,042 nonzeros. The optimal bipartitionings and corresponding communication volumes are made publicly available in a benchmark collection.

© 2015 Elsevier Inc. All rights reserved.

1. Introduction

Parallel iterative linear system solvers can be tremendously accelerated by using a good partitioning of the sparse matrices involved, which means that the parts have nearly equal size and that there are few dependencies between them. Other parallel computations that are based on sparse matrix–vector multiplication (SpMV), such as eigensystem solvers, can benefit as well.

The *sparse matrix partitioning problem* can be defined as finding a partitioning of a sparse $m \times n$ matrix A with N nonzeros in p disjoint parts,

$$A = \bigcup_{i=0}^{p-1} A_i, \qquad (1)$$

such that the number of nonzeros of part A_i satisfies

$$|A_i| \le (1+\varepsilon)\left\lceil \frac{N}{p} \right\rceil, \quad \text{for } 0 \le i < p, \qquad (2)$$

where $\varepsilon \ge 0$ is a given load-imbalance parameter, and such that the communication volume in the corresponding parallel SpMV is minimized. The load balance constraint (2) is formulated such that the extreme case $\varepsilon = 0$ still has a feasible solution. The *communication volume* of a matrix column j is defined as $\lambda_j - 1$, where λ_j is the number of matrix parts with a nonzero in column

* Corresponding author.
E-mail addresses: d.m.pelt@cwi.nl (D.M. Pelt), R.H.Bisseling@uu.nl (R.H. Bisseling).

http://dx.doi.org/10.1016/j.jpdc.2015.06.005
0743-7315/© 2015 Elsevier Inc. All rights reserved.

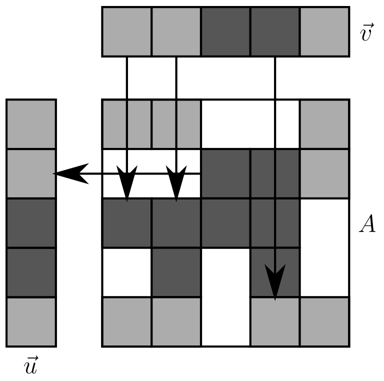

Fig. 1. Parallel multiplication of a 5×5 sparse matrix A and a dense input vector $\vec{v}$ giving a dense output vector $\vec{u} = A\vec{v}$. The 16 nonzero elements of A have been partitioned and assigned to $p = 2$ processors, depicted in light and dark gray. The vector components have also been assigned to these two processors. The parallel computation starts by communicating three vector components v_j, as depicted by vertical arrows, then it computes and adds all products $a_{ij}v_j$ locally, and finally it sends one contribution, for the second row of A, as depicted by a horizontal arrow, to enable computation of the output components $u_i = \sum_i a_{ij}v_j$. Note that the other rows do not require communication. The total communication volume is $Vol = 4$. The load balance of the nonzeros is perfect ($\varepsilon = 0$).

j. This volume occurs because in a parallel SpMV,

$$\vec{u} = A\vec{v}, \tag{3}$$

we have to send input vector component v_j to all parts that have a nonzero in column j, except for one part, provided we assign vector component v_j to one of the λ_j parts. This communication is shown as vertical arrows in Fig. 1. Similarly, we can define the communication volume of a matrix row. The total communication volume $Vol = Vol(A_0, \ldots, A_{p-1})$ is then the sum of the communication volumes of all rows and columns, and our optimization objective is to minimize Vol.

Finding an optimal sparse matrix partitioning is NP-hard, even for $p = 2$, because the underlying hypergraph partitioning problem is NP-hard [31]. Therefore, most solution methods so far have been heuristic, trying to find a good but not necessarily optimal partitioning in reasonable time. An example of a fast heuristic is the medium-grain method [34] which we recently developed; this method will be briefly explained in Section 3.

The purpose of the present article is to find an optimal solution, accepting much longer computation times, and if needed limiting the size of the problems we can solve. Our motivation is that having a suite of problems with optimal partitionings will be useful, because we can then compare heuristic solutions with an optimal benchmark solution, and see how good the heuristic methods really are. As heuristics are improving, perhaps even to the point of saturation, it may be beneficial to know how far we are from an optimal solution, and perhaps decide to optimize further for other, secondary objectives instead, such as the total number of messages sent.

In this article, we will concentrate on bipartitioning, i.e. $p = 2$, because this is the easiest problem and we can expect to build a larger suite of solved problems than for $p > 2$, and also because many partitioners are based on recursive bipartitioning. The resulting suite will be made available through the website of the Mondriaan package.[1] Furthermore, we present in this article and on the website a set of pictures of optimal solutions for small matrices. In our experience, visualization of optimal partitionings is not only pleasing to the eye, but also helpful in inspiring new ideas for improving current heuristic solution methods. As a matter of fact, this is how we were led to design the medium-grain method [34].

[1] http://www.staff.science.uu.nl/~bisse101/Mondriaan/.

2. Related work

Çatalyürek and Aykanat [10] were the first to formulate the minimization of the communication volume of a parallel SpMV as a hypergraph partitioning problem, thus solving the problem in the correct metric. Previously, graph partitioning was commonly employed, which only gives an approximation of the correct volume. In the *row-net model* of Çatalyürek and Aykanat, the n columns of the sparse matrix are modeled by the vertices of a hypergraph, and the m rows are modeled by nets (hyperedges, i.e. subsets of the vertices), such that vertex i is contained in net j if and only if $a_{ij} \neq 0$. The balance criterion of Eq. (2) is translated into a criterion on the weights of the vertices, where the weight of vertex j is defined as the number of nonzeros in matrix column j. The communication volume is modeled as the sum of the costs $\lambda_i - 1$ for all nets (rows) i. In the *column-net model*, the roles of rows and columns are reversed. Both models yield a one-dimensional (1D) matrix partitioning.

A different model by the same authors is the *fine-grain model* [11], which is two-dimensional (2D) in nature. It models the N nonzeros as vertices in a hypergraph and it has both m row nets and n column nets, defined similarly as in the 1D case. This model also minimizes the correct volume, and since it is more general it can in principle achieve better solutions; this is at the cost of longer computation times and more memory usage, as the hypergraph has many more vertices.

A different 2D method for $p > 2$ can be obtained by repeatedly bipartitioning a submatrix of A, trying both 1D hypergraph models with $p = 2$, and using the best of the two, which is done in the earlier versions of the Mondriaan package [38] (until version 3), and which we call the *localbest* method. In the latest version (version 4) of Mondriaan, the default has been changed to the recent *medium-grain* method.

Communication volume may not be the only relevant metric for the actual communication time of a parallel SpMV. Boman, Devine, and Rajamanickam [6] present a 2D method based on combining 1D graph/hypergraph partitioning with a 2D block distribution that also limits the total number of messages, besides trying to minimize the communication volume. A different approach to minimize other metrics as well is taken by the authors of the UMPa package [13], where the total and maximum volume per processor, and the total and maximum number of messages per processor can be chosen as primary or secondary objectives, and the secondary objective is used to break ties. This necessitates the use of a *directed hypergraph*, where every net has a source vertex.

Several software packages for hypergraph partitioning are currently available: sequential packages hMetis [25], PaToH [10], Mondriaan [38], and the parallel packages Parkway [37] and Zoltan [17]. Zoltan also contains a parallel toolkit Isorropia [5] that provides a sparse matrix partitioning interface (currently only for 1D partitioning). All these partitioners are heuristic, and all are based on a multilevel approach, first coarsening the hypergraph to obtain a smaller hypergraph that still resembles the original one, then obtaining an initial partitioning, and finally projecting back the solutions during the uncoarsening, while further refining them.

Graph partitioners have been studied for at least four decades, with the seminal paper by Kernighan and Lin [28] providing one of the first heuristic algorithms. The graph partitioning problem is usually defined as partitioning the vertices of a graph in such a way that the number of vertices is balanced with an allowed imbalance fraction of ε, similar to Eq. (2), and the objective is to minimize the total *edge cut*, the number of edges of the graph with vertices in different parts of the partitioning. Kernighan and Lin proposed a bipartitioning procedure which starts with a random partitioning, and then repeatedly swaps a pair of vertices between the two parts, choosing the swap with the largest possible gain, irrespective of

Download English Version:

https://daneshyari.com/en/article/431437

Download Persian Version:

https://daneshyari.com/article/431437

Daneshyari.com