Block partitioning of sparse rectangular matrices

Andrei Dumitrasc1,*, Philippe Leleux2, and Ulrich Rüde1,2

1 Friedrich-Alexander-Universität Erlangen-Nürnberg, Chair of Computer Science 10 (System Simulation), Cauerstraße 11, 91058 Erlangen, Germany
2 Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique, Parallel Algorithms Team, 42 Avenue Gaspard Coriolis, 31057 Toulouse Cedex 1, France

Received: 20 May 2019 Accepted: 6 June 2019
DOI: 10.1002/pamm.201900287

We present a means of reordering large, sparse rectangular matrices such that their nonzeros are closer to the diagonal. This enables a block-partitioning which is useful in parallel contexts. We use the Reverse Cuthill-McKee (RCM) algorithm on the adjacency matrix of the associated bipartite graph. The resulting, reordered matrix has a block bidiagonal structure.

© 2019 The Authors Proceedings in Applied Mathematics & Mechanics published by Wiley-VCH Verlag GmbH & Co. KGaA Weinheim

1 Introduction

In large scale simulations, sparse matrices appear frequently as the result of discretizations of the underlying PDE. The associated system of linear equations can be solved using the Augmented Block Cimmino Distributed method [2] which adds columns to the matrix such that the row-blocks are mutually orthogonal. To minimize the number of these additions, it is important to have the nonzeros close to the diagonal. We will describe a procedure to achieve this goal.

Let $A \in \mathcal{M}_{m \times n}$ be a sparse rectangular matrix. We can attach to it the following undirected bipartite graph $G = (E, R, C)$, with $R = \{1, 2, ..., m\}$ the set of nodes denoting row indices, $C = \{1, 2, ..., n\}$ the set of nodes denoting column indices and $E = \{(i,j) | A_{i,j} \neq 0, i \in R, j \in C\}$ the set of edges, corresponding to matrix nonzeros. We note that the following considerations assume the graph is connected. If that is not the case, the procedure can be applied to every connected component of the graph.

The same information can be represented by the graph $\mathcal{G} = (\mathcal{E}, \mathcal{V})$ with $\mathcal{V} = \{1, 2, ..., m, m + 1, m + 2, ..., m + n\}$, the set of nodes denoting both row and column indices, i.e. $\mathcal{V} = R \cup C$, $C = \{c + m, c \in C\}$ and $\mathcal{E} = \{(i,j) | i \in R, j = i + m, j \in C, A_{i,j} \neq 0\}$ the set of edges. The adjacency matrix for $\mathcal{G}$ is the symmetric matrix

$$\hat{A} = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}. \quad (1)$$

2 Cuthill-McKee

We seek to reorder $A$ such that its nonzeros are closer to the diagonal. To this end, we apply the Cuthill-McKee (CM) algorithm (see [1, Chapter 8]) to $\hat{A}$ and use the result to reorder $A$, thus reducing its bandwidth, and giving it a particular structure. Reversing this order leads to Reverse Cuthill-McKee, a variant of the algorithm where the matrix is better suited for direct solving, as it generates less fill-in.

The CM algorithm works by relabeling the nodes of an undirected graph, so it can be used on the associated, symmetric adjacency matrix. This algorithm can be thought of as a particular form of Breadth First Search, where the neighbors of each node are visited according to an increasing order of their degree, and the starting node is the one having the minimum degree. Visiting a node’s neighbors implies relabeling them with the smallest unused label. For example, given a node $i$ with neighbors $p$, $t$, $q$, CM will sort them in increasing order of degree, labeling them as $i + 1$, $i + 2$, $i + 3$, if these labels are still available. A level set is the set of nodes neighboring at least one node of the previous level set. The first level set contains only the starting node.

3 CM on rectangular matrices

The particular form of adjacency defined by the bipartite graph also implies that level sets alternate between sets of row indices and sets of column indices. The reordered matrix $\hat{A}$ is $\hat{A}^R$, with every row containing indices of the relabeled neighbors of the node on the diagonal. Every level set is represented by a square block on the diagonal, of size equal to the number of nodes in the set. Without loss of generality, in what follows we will assume that the first node chosen by the CM reordering corresponds to a row index. If that is not the case, the initial matrix should be transposed before and after the procedure.

* Corresponding author: e-mail andrei.dumitrasc@fau.de, phone +49 9131 85 67291
Let $A^R$ be the matrix generated from the blocks of $\hat{A}^R$ as shown below (see also [3]), with $n_i$, the sizes of the blocks corresponding to level sets. The matrix $A^R$ is the reordered form of $A$, having its nonzeros closer to the diagonal.

$$
\hat{A}^R = \begin{bmatrix}
0 & A_{11} & A_{12}^T & 0 & A_{13}^T & 0 & \cdots \\
A_{21} & 0 & A_{22} & A_{23}^T & 0 & A_{24}^T & \cdots \\
0 & A_{32} & 0 & A_{33} & 0 & A_{34}^T & \cdots \\
& \ddots & \ddots & \ddots & \ddots & \ddots & \ddots
\end{bmatrix}
$$

$$
A^R = \begin{bmatrix}
A_{11} & A_{12} & A_{13} & A_{14} \\
A_{21} & A_{22} & A_{23} & A_{24} \\
A_{31} & A_{32} & A_{33} & A_{34} \\
& \ddots & \ddots & \ddots \\
& & & \ddots & \ddots & \ddots
\end{bmatrix}
$$

\begin{align*}
A_R &= \{n_1, n_3, n_5, n_7, \ldots \} \\
A^R &= \{n_2, n_4, n_6, n_8, \ldots \}
\end{align*}

4 Results

In Fig. 1, we show a case where we reordered a random matrix generated with Matlab. After constructing $\hat{A}$ as in Eq. 1 and applying RCM, we can see the reordered initial matrix in the 3rd image, with its blocks in red borders. Notice the significantly narrower bandwidth. In Fig. 2, we illustrate the same procedure on a matrix from the Suite Sparse Matrix Collection related to a combinatorial problem. Again, the nonzeros were brought significantly closer to the diagonal.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig1.png}
\caption{Reordering of a small, random matrix}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig2.png}
\caption{Reordering of an application matrix}
\end{figure}

5 Conclusion and future work

In the above, we have described a procedure for reordering large, sparse, rectangular matrices such that they have a structure suitable for block partitioning. We have achieved this by applying the Reverse Cuthill-McKee algorithm on the adjacency matrix of the associated bipartite graph. Although the results are encouraging, we also notice that there is a significant difference in the size of the blocks, which might have a negative impact on performance in a parallel computing setting. Thus, a potential avenue for future developments would be to find a way of obtaining similarly sized blocks. Also, it would be desirable to implement this algorithm in an optimized fashion, such that it can be used in large scale computations.

References

[1] Duff I.S., Erlisman A.M., Reid J.K., Direct methods for sparse matrices, Oxford Univ. Press, Oxford, 2017.
[2] Duff I.S., Guivarch R., Ruiz D., Zenadi M., The augmented block Cimmino distributed method, SIAM J. Sci. Comp., 37(3)(2015), A1248-A1269.
[3] Reid J. K., Scott A. J., Reducing the total bandwidth of a sparse unsymmetric matrix, SIAM J. Matrix Anal. Appl., 28(3)(2006), 805-821.

\footnote{https://sparse.tamu.edu/}