Elimination tree flattening to exploit right-hand sides sparsity*

Patrick Amestoy[†], Jean-Yves L'Excellent[‡] and Gilles Moreau[‡]

Motivations. The cost of the solve phase of sparse direct methods can be critical in applications where systems of linear equations with thousands of Right-Hand Sides (RHS) must be solved. In such applications, the RHS are often very sparse. This is for example the case of (i) seismic modeling or electromagnetism applications, where each RHS vector corresponds to a geometrically localized source; (ii) Schur complement computations, where each RHS may be a column of a sparse matrix whose nonzero geometrical locality results from the local physical interactions of the discretized problem. Efficiently exploiting RHS sparsity leads to the combinatorial problem of finding a suitable column permutation of the RHS. To address this issue, two original algorithms are proposed and experimented on two geoscience applications.

Problem Specification. We consider systems of linear equations with multiple Right-Hand Sides (RHS) $AX = B$, where A and B are $n \times n$ and $n \times m$ sparse matrices. When m is large, direct methods are often preferred because the matrix is factored only once ($A = LU$). The system can then be solved for all RHS through forward elimination ($LY = B$) followed by backward substitution ($UX = Y$). We focus on the forward elimination but the work presented here has a larger scope of application.

Although our algorithms were designed for general elimination trees, for the sake of simplicity we describe them on separator trees resulting from recursive nested dissection. The forward elimination follows a bottom-up traversal of the tree, noted T. If Δ denotes the number of operations for the forward elimination then we have $\Delta = m \times \sum_{u \in T} \delta_u$ where the number of operations δ_u at node u depends on the size of the partial L factor computed at that node.

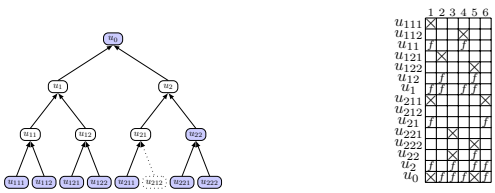

Figure 1: Each node of the separator tree (left) corresponds to a row index in B. In $L^{-1}B$ (right), $\times$ denotes an original entry in B and f a fill(new entry) in $L^{-1}B$.

Thanks to the nonzero structure of B, nodes in the tree can be skipped (node u_{212} in Figure 1) [3], leading to a pruned tree $T_p(B)$. We thus have $\Delta(B) = m \times \sum_{u \in T_p(B)} \delta_u$ – compare DENse (`DEN`) and Tree-Pruned (`TP`) in Figure 2. The notions of node intervals [1] and of column permutations based on a post-ordering have then been used to further reduce computation. At each node $u \in T_p(B)$, the number of columns is reduced from m to an interval whose size $\theta_u(\sigma)$ is given by the first and last entries in each row and depends on the permutation σ of the columns (compare the INItial (`INI`) and the Post-Order (`PO`) orderings in Figure 2). The *combinatorial problem* that we want to address here consists in finding a permutation σ of the columns such that $\Delta(B, \sigma) = \sum_{u \in T_p(B)} \delta_u \times \theta_u(\sigma)$ is minimized.

Contributions and results. Intuitively, we seek a permutation such that RHS columns with close nonzero structure (with respect to the separator tree) are close in the permutation. While `PO` decides of a column's position based on the position in the tree of one arbitrary entry in that column, our new algorithm identifies and gathers recursively RHS columns with common characteristics by scanning their nonzero structure dur-

*This extended abstract overlaps with [2], submitted to SIAM SISC. The work was partially supported by the LABEX MILYON (ANR-10-LABX-0070) of Univ. Lyon.

[†]Univ. Toulouse, INPT and IRIT laboratory, France

[‡]Univ Lyon, CNRS, ENS de Lyon, Inria, Université Claude-Bernard Lyon 1, LIP UMR5668, Lyon, France.

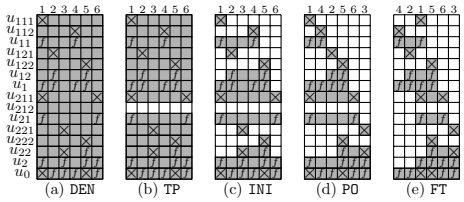

Figure 2: Exploiting sparsity in RHS structures (grey areas are treated as nonzeros).

ing a top-down traversal of $T_p(B)$. We call it the Flat Tree (`FT`) algorithm because of the analogy with the ordering that one would obtain when "flattening" the tree.

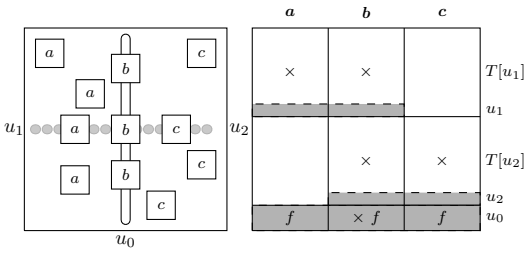

Figure 3: Illustration of the "flat tree" permutation on a 2D domain. $T[u_i]$ is the subtree of T rooted at u_i. For each submatrix, × means at least one nonzero, grey means dense, and blank means empty.

Figure 3 illustrates the first step of the algorithm on a set of RHS distributed in a 2D domain. We identify the position of each RHS column according to the current separator u_0, then form the three sets of columns $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$ and finally to permute the RHS as indicated in Figure 3. The same operation is performed recursively on a (resp. c) depending on RHS positions with respect to u_1 (resp. u_2), but also on b depending on each RHS position with respect to u_1 and u_2 (see [2] for details and for the generalization to a fully algebraic approach applicable to arbitrary RHS).

To further reduce Δ, we propose to split B into groups of submatrices with (almost) disjoint nonzero structures through a top-down traversal of the tree while creating as few groups as possible (e.g., for BLAS 3 efficiency or granularity of parallelism within each group). Given the minimal number of operations Δ_{min} obtained when columns of B are processed one by one, our algorithm decreases Δ below an arbitrary threshold $\mu_0\Delta_{min}$, where $\mu_0 = 1.01$ means Δ is within 1% of Δ_{min}. In Figure 3, submatrices $\boldsymbol{a}$ and $\boldsymbol{c}$ have disjoint structure except for node u_0 whereas $\boldsymbol{b}$ have common nonzeros with both $\boldsymbol{a}$ and $\boldsymbol{c}$ so that two groups can be created: $[\boldsymbol{a}, \boldsymbol{c}]$ and $[\boldsymbol{b}]$. In Figure 2(e), splitting B into the two groups of columns $[\{4, 2\}, \{6, 3\}]$ and $[\{1, 5\}]$ leads to $\Delta = \Delta_{min}$ after the first step of the algorithm.

Our algorithms were experimented on a set of problems from electromagnetism and seismic applications involving large numbers of *sparse* RHS (see Table 1). With `FT`, Δ is always lower than with the standard `PO` strategy. Moreover, when grouping is also used, the number of groups (NG) remains very small on all cases even when the amount of extra operations allowed compared to Δ_{min} is negligible (here 1%). We have thus shown that `FT` combined with the proposed grouping strategy correctly addresses our combinatorial problem on the applications considered.

Table 1: Δ ($\times 10^{13}$) for $LY = B$. NG: number of groups when grouping is combined with FT and $\mu_0 = 1.01$.

matrices	7Hz	H0	S3	D30
$n(\times 10^6)$	7.2	0.3	3.3	29.7
m	2302	8000	12340	3914
`DEN`	5.94	.39	13.36	71.60
`TP`	2.54	.11	4.98	39.78
`INI`	1.46	.086	3.73	19.49
`PO`	1.21	.068	2.65	10.93
`FT`	.92	.057	2.17	10.21
NG	3	4	5	4
$\Delta_{min}(\times 10^{13})$	.69	.050	1.71	7.31

References

[1] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and F.-H. Rouet, *Parallel computation of entries of A^{-1}*, SIAM J. Sci. Comput., 37 (2015), pp. C268–C284.

[2] P. R. Amestoy, J.-Y. L'Excellent, and G. Moreau, *On exploiting sparsity of multiple right-hand sides in sparse direct solvers*, Research report RR-9122, INRIA, 2017.

[3] J. R. Gilbert, *Predicting structure in sparse matrix computations*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 62–79.