A Combinatorial Problem in Sparse Orthogonal Factorization

Esmond G. Ng

Lawrence Berkeley National Laboratory

Barry W. Peyton

Dalton State College

Sparse Days, CERFACS, September 6-7, 2011

The Problem ...

Given a sparse $m \times n$ matrix A, with $m \geq n$.

Let $A = Q_A \begin{bmatrix} R_A \\ O \end{bmatrix}$ be its orthogonal factorization, where Q_A is orthogonal and R_A is upper triangular.

$$A = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & & & & \\ & \times & & & \\ & & \times & & \\ & & & \times & \\ & & & & \times \end{bmatrix} , \quad R_A = \begin{bmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \\ & & & & \times \end{bmatrix}$$

The Problem ...

Given a sparse $m \times n$ matrix A, with $m \geq n$.

Let $A = Q_A \begin{bmatrix} R_A \\ O \end{bmatrix}$ be its orthogonal factorization, where Q_A is orthogonal and R_A is upper triangular.

$$A = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & & & & \\ & \times & & & \\ & & \times & & \\ & & & \times & \\ & & & & \times \end{bmatrix} , \quad R_A = \begin{bmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \\ & & & & \times \end{bmatrix}$$

The Problem ...

$\boxed{\text{Extract a } k \times n \text{ submatrix } C \text{ from } A\text{, with } k < m.}$

Suppose $C = Q_C \begin{bmatrix} R_C \\ O \end{bmatrix}$, where Q_C is orthogonal and R_C is upper triangular.

Desirable properties of C:

- R_C is "close" to R_A
- R_C is much sparser than R_A

The Problem ...

$\boxed{\text{Extract a } k \times n \text{ submatrix } C \text{ from } A\text{, with } k < m.}$

Suppose $C = Q_C \begin{bmatrix} R_C \\ O \end{bmatrix}$, where Q_C is orthogonal and R_C is upper triangular.

Desirable properties of C:

- R_C is "close" to R_A
- R_C is much sparser than R_A

Example ...

$$A = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & & & & \\ & \times & & & \\ & & \times & & \\ & & & \times & \\ & & & & \times \end{bmatrix}, \quad C = \begin{bmatrix} \times & & & & \\ & \times & & & \\ & & \times & & \\ & & & \times & \\ & & & & \times \end{bmatrix}$$

Objective ...

We want to move as many rows as possible from A to C so that R_C is close to R_A, while ensuring that R_C is as sparse as possible so that the cost of computing and storing R_C is small.

That is, we are partitioning A into

$$A = \begin{bmatrix} C \\ D \end{bmatrix}$$

where C is the "sparse" portion of A and D is the "dense" portion of A.

This is a purely *symbolic* process. We do not take numerical values into consideration.

Why is C useful?

Solution of sparse least squares problems ...

$$\min_z \|Az - b\|_2$$

Useful in both direct and iterative solutions.

Direct solution of sparse least squares ...

- Suppose we have the QR factorization of C.
- Then the least squares problem can be written as

$$\min_z \|Az - b\|_2 = \min_z \left\| \begin{bmatrix} C \\ D \end{bmatrix} z - \begin{bmatrix} c \\ d \end{bmatrix} \right\|_2$$

where D may be considered to contain the “dense” rows of A.

- Solve $\min_w \|Cw - c\|_2$.
- R_C, D, and d can be used to update w to obtain the solution to the original least squares problem.
- [Heath, 1982], [Bjorck, 1984], [Ng, 1991]

Direct solution of sparse least squares ...

- Suppose we have the QR factorization of C.
- Then the least squares problem can be written as

$$\min_z \|Az - b\|_2 = \min_z \left\| \begin{bmatrix} C \\ D \end{bmatrix} z - \begin{bmatrix} c \\ d \end{bmatrix} \right\|_2$$

where D may be considered to contain the "dense" rows of A.

- Solve $\min_w \|Cw - c\|_2$.
- R_C, D, and d can be used to update w to obtain the solution to the original least squares problem.
- [Heath, 1982], [Bjorck, 1984], [Ng, 1991]

Iterative solution of sparse least squares ...

- Let R_C be the upper triangular factor in the QR factorization of C.
- R_C can be used as a preconditioner

$$\min_z \|Az - b\|_2 = \min_z \|(AR_C^{-1})(R_C z) - b\|_2$$

- If D contains k rows, then LSQR converges in no more than k iterations (in theory).
- [Avron, Ng, Toledo, 2009]

Iterative solution of sparse least squares ...

- Let R_C be the upper triangular factor in the QR factorization of C.
- R_C can be used as a preconditioner

$$\min_z \|Az - b\|_2 = \min_z \|(AR_C^{-1})(R_C z) - b\|_2$$

- If D contains k rows, then LSQR converges in no more than k iterations (in theory).
- [Avron, Ng, Toledo, 2009]

A Typical Approach for Finding C ...

- C and D contain the “sparse” and “dense” rows of A, respectively.

- Order rows of A in increasing number of nonzeros.
- Let D be the last k rows.
- Thus, C contains the first $n - k$ rows of A.

- It does not take the structure of the rows into account.

- Our goal is to develop alternative heuristics in finding C by taking the sparsity structure of the rows of A into consideration.

Preliminaries ...

- A full rank $\Rightarrow A^T A$ symmetric positive definite.
  - Each row of A induces a dense submatrix in $A^T A$.
  - R_A is mathematically the Cholesky factor of $A^T A$; i.e., $A^T A = R_C^T R_C$.

- Notation:
  - $M[i, j]$ is the (i, j)-element of the matrix A.
  - $M[i, *]/M[*, i]$ is the set of nonzero elements in row/column i of M.
  - Struct(v) is the set of indices of the nonzero elements in row/column vector v.

- Associated with R_A is an elimination tree $T(R_A)$ (or, simply, $T(A)$)
  - There is an edge between vertices x_i and x_j ($j > i$) iff $R_A[i,j]$ is the first off-diagonal nonzero in row i of R_A.
  - Level(x_i) $=$ length of path joining x_i and the root in $T(R_A)$.

- If $R_A[i,j] \neq 0$ ($i \leq j$), then x_j is an ancestor of x_i in $T(R_A)$.

- Let $A[i, f_i]$ be the *first* nonzero element in row i of A. For $f_i \leq k \leq n$, if $A[i,k]$ is nonzero, then x_k must be an ancestor of x_{f_i} in $T(R_A)$, since $R_A[f_i, k]$ must be nonzero.

Heuristics ...

We will look at several heuristics for finding C ...

Heuristic #1 ...

- C is initially empty.
- Rows are moved from A to C one at a time.
- Suppose k rows have been moved.
- Row i of A will be moved next if

$$\text{Struct}(A[i,*]) \subseteq \text{Struct}(R_C[f_i,*])$$

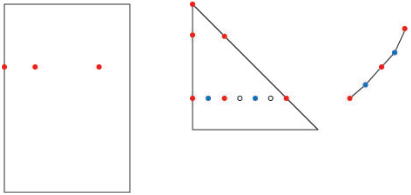
- If no rows satisfy the condition above, then row i will be moved if $|\text{Struct}(A[i,*])|$ is the smallest.
  - Ties are broken arbitrarily.

Heuristic #2 ...

- Suppose that

$$\mathrm{Struct}(A[i,*]) = \{j_1, j_2, \cdots, j_{t_i}\}$$

  with $f_i = j_1 < j_2 < \cdots < j_{t_i}$.
- Row i induces a $t_i \times t_i$ dense submatrix in $A^T A$, with $\mathrm{Struct}(A[i,*])$ as the set of row/column indices.
- Consider row j_s of R_A^T, $1 < s < t_i$. This row must contain nonzeros in column p, where x_p is a vertex along the path joining x_{f_i} and x_{j_s} in $T(R_A)$.

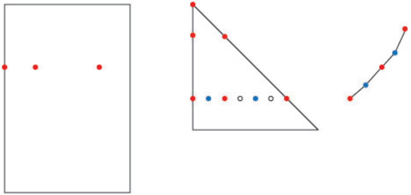


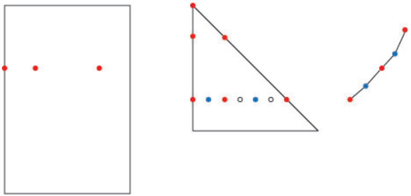
Heuristic #2 ...

- Suppose that

$$\text{Struct}(A[i,*]) = \{j_1, j_2, \cdots, j_{t_i}\}$$

  with $f_i = j_1 < j_2 < \cdots < j_{t_i}$.
- Row i induces a $t_i \times t_i$ dense submatrix in $A^T A$, with Struct$(A[i,*])$ as the set of row/column indices.
- Consider row j_s of R_A^T, $1 < s < t_i$. This row must contain nonzeros in column p, where x_p is a vertex along the path joining x_{f_i} and x_{j_s} in $T(R_A)$.





- What if row i A is discarded?
  - Level(f_i) − Level(j_s) is an upper bound on the number of nonzeros removed in row j_s of R_A.
  - Summing Level(f_i) − Level(j_s) over s gives an upper bound on the number of nonzeros that would be removed from R_A.

Heuristic #2 ...

- C is initially the same as A.
- Suppose that

$$\text{Struct}(A[i,*]) = \{j_1, j_2, \cdots, j_{t_i}\}$$

with $f_i = j_1 < j_2 < \cdots < j_{t_i}$.

- At step k, remove row i from C if

$$\sum_{s=1}^{t_i} [\text{Level}(f_i) - \text{Level}(j_s)]$$

is maximized.

- Level is applied to the current R_C.

Heuristic #3 ...

- C is initially empty.
- Suppose k rows have been moved from A to C.

- Consider one of the remaining rows, say, row i of A, and suppose that $\text{Struct}(A[i,*]) = \{j_1, j_2, \cdots, j_{t_i}\}$.
- Associate a tree (forest) T_i with row i of A:
  - T_i includes every x_{j_s} and *all* the ancestors of x_{j_s} (in $T(R_C)$), for $1 \leq s \leq t_i$.
  - Denote the set of *leaves* of T_i by $\text{Leaves}(i) = \{\ell_1, \ell_2, \cdots, \ell_{r_i}\}$.
  - For each leaf ℓ_q, define

    $$d_{i,q} = |\{p : p > \ell_q \text{ and } p \in \text{Struct}(A[i,*]) \setminus \text{Struct}(R_C[\ell_q,*])\}|$$

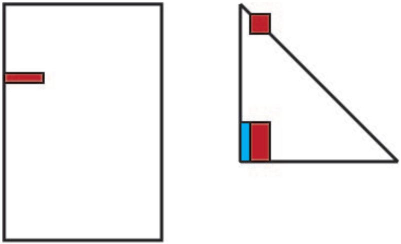
  - Define $\text{Fill}(i,q) = d_{i,q}|\text{Struct}(R_C[\ell_q,*])| + \frac{1}{2}d_{i,q}(d_{i,q}-1)$.
  - Associate a weight with $A[i,*]$: $\text{Wt}(i) = \sum_{q=1}^{r_i} \text{Fill}(i,q)$

- Move row i from A to C if $\text{Wt}(i)$ is maximized.

Heuristic #3 ...

$$d_{i,q} = |\{p : p > \ell_q \text{ and } p \in \mathrm{Struct}(A[i,*]) \setminus \mathrm{Struct}(R_C[\ell_q,*])\}|$$

$$\mathrm{Fill}(i,q) = d_{i,q}|\mathrm{Struct}(R_C[\ell_q,*])| + \frac{1}{2} d_{i,q}(d_{i,q}-1)$$



Heuristic #4 ...

- Similar to Heuristic #3, with the exception that $\mathrm{Wt}(i)$ is a weighted sum of $\mathrm{Fill}(i,q)$:

$$\mathrm{Wt}(i) = \sum_{q=1}^{r_i} \mathrm{Fill}(i,q)[\mathrm{Level}(\ell_q) - \mathrm{Level}(\mathrm{lca}(\ell_q, \ell_{q+1}))]$$

  where $\mathrm{lca}(\ell_q, \ell_{q+1})$ is the *least commmon ancestor* of ℓ_q and ℓ_{q+1}.
- The effect is to penalize row i of A if it has large bushy subtrees associate with it.

A Couple of Examples ...

- Examples from the University of Florida Sparse Matrix Collection.
- Matrices are $n \times m$, with $m \geq n$. We transpose the matrices in the experiments.
- Columns of each (transposed) matrix are preordered using a variant of the minimum degree algorithm.
  - Columns of A (after some rows have been removed) may be reordered again.
- All 4 heuristics were tested.

- Transpose of lp_nug12: $m = 8,856$, $n = 3,192$, $|A| = 38,304$.

rows	#4	#3	#2	#1
0	2,838,887	2,590,922	2,777,024	2,537,090
5	2,661,364	2,564,994	2,761,537	2,603,586
10	3,021,141	2,545,864	2,554,842	2,538,128
15	2,848,972	2,496,094	2,550,362	2,492,292
20	2,755,986	2,489,869	2,547,968	2,486,739
25	2,848,584	2,438,093	2,537,020	2,411,402
30	2,673,133	2,607,836	2,531,279	2,463,227
35	2,695,931	2,449,566	2,526,185	2,408,312
40	2,896,024	2,968,592	2,521,931	2,196,613

Example #2 ...

- Tranpose of dano3mip: $m = 15,851$, $n = 3,202$, $|A| = 81,633$.

rows	#4	#3	#2	#1
0	1,842,580	3,405,327	1,842,580	3,396,040
5	1,088,871	1,165,385	1,585,636	1,113,808
10	1,097,843	1,094,251	1,088,557	1,077,338
15	1,080,292	1,085,694	1,088,078	1,083,113
20	1,102,819	1,033,285	1,087,567	1,082,760
25	1,161,665	1,019,705	1,082,563	1,082,418
30	1,126,726	1,022,377	1,082,218	1,078,359
35	1,074,726	1,022,036	1,081,809	1,078,019
40	1,058,418	1,031,597	1,081,366	1,078,321

Remarks ...

- We investigated a number of heuristics for removing rows in sparse orthogonal factorization.
  - Very preliminary work ...
- Simple heuristics do not always work well. More sophisticated schemes seem to be needed.
- Related research problems:
  - updating the elimination tree when the matrix is changed
  - updating the column ordering when the matrix is changed

Remarks ...

- We investigated a number of heuristics for removing rows in sparse orthogonal factorization.
  - Very preliminary work ...
- Simple heuristics do not always work well. More sophisticated schemes seem to be needed.
- Related research problems:
  - updating the elimination tree when the matrix is changed
  - updating the column ordering when the matrix is changed