Does the TSP heuristic for minimizing block counts in sparse Cholesky factorization have to be expensive?

Esmond G. Ng
Lawrence Berkeley National Laboratory

Mathias Jacquelin
Lawrence Berkeley National Laboratory

Barry W. Peyton
Dalton State College

Sparse Days @ CERFACS
November 23-24, 2020

What we know ...

- Let L be the Cholesky factor of a large, sparse, symmetric, positive definite matrix A.
- When A is ordered appropriately, blocks of consecutive columns in L called *supernodes* have essentially identical sparsity structure.
- Within each supernode, a row below the diagonal block is either entirely nonzero or entirely zero.
- A **dense block** is a set of consecutive nonzero rows in a supernode.
- Reordering columns within a supernode does not change the number of nonzeros in L, but it changes the number of dense blocks.

$$L = \left[\begin{array}{cc|cc|ccccc} 1 & & & & & & & & \\ * & 2 & & & & & & & \\ \hline & & 3 & & & & & & \\ & & * & 4 & & & & & \\ \hline * & * & * & * & 5 & & & & \\ * & * & & & * & 6 & & & \\ & & * & * & * & * & 7 & & \\ * & * & & & * & * & * & 8 & \\ & & * & * & * & * & * & * & 9 \end{array}\right] \qquad \hat{L} = \left[\begin{array}{cc|cc|ccccc} 1 & & & & & & & & \\ * & 2 & & & & & & & \\ \hline & & 3 & & & & & & \\ & & * & 4 & & & & & \\ \hline * & * & & & 6 & & & & \\ * & * & & & * & 8 & & & \\ * & * & * & * & * & * & 5 & & \\ & & * & * & * & * & * & 7 & \\ & & * & * & * & * & * & * & 9 \end{array}\right]$$

- L has a *supernodal elimination tree* T.
- The nonzeros in row i of L form a subtree rooted at i's supernode in T, which is called a *row subtree* and is denoted by $T_r[i]$.

The TSP Problem

- **[Pichon, Faverge, Ramet, and Roman (2017)]** wanted to reorder the columns within each supernode to minimize the number of dense blocks linking one supernode with another.
- Pichon et al. translated the problem into a set of *traveling salesman problems* (TSP), one for each supernode.
- They solved each instance of TSP using an effective *insertion method* [Rosenkrantz, Stearns, and Lewis II (1977)].

- **Strength:** Very good at reducing the number of dense blocks [Pichon et al. (2017); Jacquelin, Ng, and Peyton (2018)].
- **Weakness:** High cost in time [Pichon et al. (2017); Jacquelin et al. (2018)].
- **Primary bottleneck:** Computing the "distances".

Computing distances in the TSP's

- Consider supernode S. Let i and j be two columns in S.
- The *distance* between i and j is defined by

$$d^S(i,j) = |T_r[i] \setminus T_r[j]| + |T_r[j] \setminus T_r[i]| \,.$$

- **Straightforward approach:** Computing the distances requires $|S|$ traversals, member-by-member, of each set $T_r[i]$.
  - Pichon et al. used this straightforward approach (**Bordeaux**).
- **Two observations that lead to two improvements:**
  1. $d^S(i,j) = 0$ **if and only if** $T_r[i] = T_r[j]$.
     - Columns of S are partitioned into equivalence classes. Only a representative from each class is included in the TSP (**Reduction**).
  2. $d^S(i,j) = |T_r[i]| + |T_r[j]| - 2|T_r[j] \cap T_r[i]|$.
     - [Gilbert, Ng, and Peyton (1994)] provided an efficient way to compute the size of a subtree by partitioning the edges into disjoint paths (**Paths**).

Experiments

- Took 31 symmetric matrices from the SuiteSparse Matrix Collection, with dimension $\geq 500,000$.
- Compared times for the Bordeaux TSP code and our TSP improvements.
- Evaluated the timings in the context of the symbolic factorization (SF) in which they occur.