Recent Results on Improving Performance of Sparse Cholesky Factorization by Reordering Columns within Supernodes

M. Ozan Karsavuran, Esmond G. Ng, Barry W. Peyton

Abstract

Let A be an n by n sparse symmetric positive definite matrix, and let $A = LL^T$ be the Cholesky factorization of A, where L is a lower triangular matrix. It is well known that L suffers *fill* during such a factorization; that is, L will have nonzero entries in locations occupied by zeros in A. As a practical matter, it is important to limit the number of such fill entries in L. Consequently, software for solving a sparse symmetric positive definite linear system $Ax = b$ via sparse Cholesky factorization requires the following four steps.

First, compute a fill-reducing ordering of A using either the *nested dissection* [5, 11] or the *minimum degree* [1, 6, 12, 15] ordering heuristic (the **ordering** step). Second, compute the needed information concerning and data structures for the sparse Cholesky factor matrix (the **symbolic factorization** step). Third, compute the sparse Cholesky factor within the data structures computed during the symbolic factorization step (the **numerical factorization** step). Fourth, solve the linear system by performing in succession a sparse forward solve and a sparse backward solve using the sparse Cholesky factor and its transpose, respectively (the **solve** step).

The authors of this work, along with J. L. Peyton, presented a thorough look [10] at some serial algorithms for the third step in the solution process (the numerical factorization step). Our goal was to improve the performance of serial sparse Cholesky factorization algorithms on multicore processors when only the multithreaded BLAS are used to parallelize the computation. Essentially, our first paper [10] explored what can be done for serial sparse Cholesky factorization using the techniques and methodology used in LAPACK.

Our primary contribution in [10] is the factorization method that we called *right-looking blocked* (RLB). Like all of the other factorization methods studied in [10], RLB relies on *supernodes* to obtain efficiency, where *supernodes* are, roughly speaking, sets of consecutive columns in the factor matrix sharing the same zero-nonzero structure. RLB, however, is unique among the factorization methods studied in [10] in that it requires *no floating-point working storage or assembly operations*; that is, the computation is performed in place within the data structures computed for the factor matrix during the symbolic factorization step. RLB is also unique among the factorization methods studied in [10] in that it is *entirely* dependent for efficiency on the existence of few and large dense blocks joining together pairs of supernodes in the factor matrix. Furthermore, the number of and size of these dense blocks are *crucially* dependent on how the columns of the factor matrix are ordered *within* supernodes. As a result, RLB is perfectly suited for studying the *quality* of algorithms for reordering columns within supernodes. It is precisely a study of this sort that will occupy our attention in this work. It should be noted that reordering the columns (and the corresponding rows) within each supernode does not change the number of nonzeros in the Cholesky factor.

Pichon, Faverge, Ramet, and Roman [14] were the first to take seriously the problem of reordering columns within supernodes, in that they were the first to treat it in a highly technical manner. They ingeniously formulated the underlying optimization problem as a *traveling salesman problem*, for which there exist powerful and effective heuristics. We will refer to their approach as TSP. The problem with their approach was not ordering quality; it was the cost, in time, of computing the needed TSP distances [8, 14]. In 2021, Jacquelin, Ng, and Peyton [9] devised a much faster way to compute the needed distances, which greatly reduces the runtimes for the TSP method.

In 2017, Jacquelin, Ng, and Peyton [8] proposed a simpler heuristic for reordering columns within supernodes based on *partition refinement* [13]. In their paper, they report faster runtimes for their method than TSP, while obtaining similar ordering quality. We will refer to their method as PR.

In this work, we perform a careful comparison of TSP and PR; we compare them, primarily, by measuring the impact of TSP and PR on RLB factorization times using Intel's MKL multithreaded BLAS on 48 cores of our test machine. This approach is justifiable since, as alluded to above, the performance of RLB depends on the quality of the TSP or PR reorderings.

The comparisons are conducted using a set of large matrices from the SuiteSparse collection [3]. In our experiments, certain small supernodes are merged together to create a coarser supernode partition. This idea was first introduced by Ashcraft and Grimes [2], and was demonstrated to reduce the factorization time at the expense of a relatively small increase in the size of the data structures. Merging supernodes has become a standard practice in software for sparse symmetric factorization, such as MA57 [4] and MA87 [7].

In this presentation, we will describe two techniques for improving the quality of the TSP reorderings; we will show that the best results for TSP are obtained when the two techniques are combined. We will also introduce a new way to reorganize the PR reordering algorithm to make it much more time and storage efficient. In addition, we will introduce a single technique for modestly improving the quality of the PR reorderings. We will further show that the enhanced PR and enhanced TSP produce orderings of virtually equal quality. However, the former requires significantly less storage to implement and runs much faster than the latter.

References

[1] Patrick R. Amestoy, Timothy A. Davis, and Iain S. Duff. An approximate minimum degree ordering algorithm. *SIAM J. Matrix Anal. Appl.*, 17(4):886–905, 1996.

[2] Cleve C. Ashcraft and Roger G. Grimes. The influence of relaxed supernode partitions on the multifrontal method. *ACM Trans. Math. Softw.*, 15(4):291–309, 1989.

[3] Timothy A. Davis and Yifan Hu. The University of Florida sparse matrix collection. *ACM Trans. Math. Softw.*, 38(1):1–28, 2011.

[4] Iain S. Duff. MA57—a code for the solution of sparse symmetric definite and indefinite systems. *ACM Trans. Math. Softw.*, 30:118–144, 2004.

[5] Alan George. Nested dissection of a regular finite element mesh. *SIAM J. Numer. Anal.*, 10(2):345–363, 1973.

[6] Alan George and Joseph W. H. Liu. The evolution of the minimum degree ordering algorithm. *SIAM Rev.*, 31(1):1–19, 1989.

[7] Jonathan D. Hogg, John K. Reid, and Jennifer A. Scott. Design of a multicore sparse Cholesky factorization using DAGs. *SIAM J. Sci. Comput.*, 32(6):3627–3649, 2010.

[8] Mathias Jacquelin, Esmond G. Ng, and Barry W. Peyton. Fast and effective reordering of columns within supernodes using partition refinement. In *2018 Proceedings of the Eighth SIAM Workshop on Combinatorial Scientific Computing*, pages 76–86, 2018.

[9] Mathias Jacquelin, Esmond G. Ng, and Barry W. Peyton. Fast implementation of the Traveling-Salesman-Problem method for reordering columns within supernodes. *SIAM J. Matrix Anal. Appl.*, 42(3):1337–1364, 2021.

[10] M. Ozan Karsavuran, Esmond G. Ng, Barry W. Peyton, and Jonathan L. Peyton. Some new techniques to use in serial sparse Cholesky factorization algorithms. Submitted to TOMS, 2024.

[11] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1999.

[12] Joseph W. H. Liu. Modification of the minimum-degree algorithm by multiple elimination. *ACM Trans. Math. Softw.*, 11(2):141–153, 1985.

[13] Robert Paige and Robert E. Tarjan. Three partition refinement algorithms. *SIAM J. Comput.*, 16(6):973–989, 1987.

[14] Gregoire Pichon, Mathieu Faverge, Pierre Ramet, and Jean Roman. Reordering strategy for blocking optimization in sparse linear solvers. *SIAM J. Matrix Anal. Appl.*, 38(1):226–248, 2017.

[15] W. F. Tinney and J. W. Walker. Direct solution of sparse network equations by optimally ordered triangular factorization. *Proc. IEEE*, 55:1801–1809, 1967.