On Evaluating Parallel Sparse Cholesky Factorizations

Wen-Yang Lin and Chuen-Liang Chen
Department of Computer Science and Information Engineering
National Taiwan University
Taipei, TAIWAN
e-mail: clchen@csie.ntu.edu.tw

Abstract

Though many parallel implementations of sparse Cholesky factorization with the experimental results accompanied have been proposed, it seems hard to evaluate the performance of these factorization methods theoretically because of the irregular structure of sparse matrices. This paper is an attempt to such research. On the basis of the criteria of parallel computation and communication time, we successfully evaluate four widely adopted Cholesky factorization methods, including column-Cholesky, row-Cholesky, submatrix-Cholesky and multifrontal. The results show that the multifrontal method is superior to the others.

1. Introduction

In the directive solution of a sparse symmetric and positive definite linear system $Ax = b$, Cholesky factorization has acted as a synonym of the decomposition of A into LL^T, for L a lower triangular. As the time-consuming characteristic of Cholesky factorization and the advance of parallel computer architectures, a demand for efficient parallel sparse Cholesky factorization algorithms have emerged.

Though a variety of parallel sparse Cholesky factorizations have been proposed [7] and usually accompanied with some experimental results, it still lacks of theoretical evaluation due to the irregularity of sparse matrices. The evaluation not only serves as a prediction of the actual performance, but also exploits the potentiality and limitation of a Cholesky factorization; thus helps in choosing the appropriate method for a particular machine. This paper is an attempt to such study. On the basis of the criteria of parallel computation and communication time, we consider four widely adopted Cholesky factorization methods, including *column-Cholesky*, *row-Cholesky*, *submatrix-Cholesky* and *multifrontal*, on distributed-memory multiprocessors. With some graph-theoretic techniques we have succeed in the evaluation and found that multifrontal method is superior to the others.

The remainder of this paper is organized as follows. In Section 2, we review some graph-theoretic notions used in matrix computation and the four Cholesky factorizations. In Section 3, we define the parallel computation and communication criteria, and formulate them in terms of graph notations. On the basis of the criteria, we evaluate the performance of these four Cholesky factorizations in Section 4. Experiments on some practical sparse matrices are also included. Section 5 gives the conclusions.

2. Background

2.1. Graph model for Cholesky factorizations

In this subsection, we briefly review the combinatorial aspect of the elimination process as formulated by Rose [12] and then confine it to a more specific graph model on which later derivations will be based.

Let $A = (a_{ij})$ be an $n \times n$ sparse symmetric positive definite matrix. The associated graph of A, $G_A = (V_A, E_A)$, is constructed as below:

$V_A = \{ v_i \mid v_i$ corresponds to column/row i of A, $1 \le i \le n \}$,

$E_A = \{ (v_i, v_j) \mid a_{ij} \ne 0,\ 1 \le i < j \le n \}$.

The symmetric Gaussian elimination of A can be described as follows. Letting

$$A = A^{(0)} = \begin{bmatrix} d & r^T \\ r & \overline{H} \end{bmatrix},$$

where d is a positive scalar, r is an $(n-1)\times 1$ and $\overline{H}$ is an $(n-1)\times(n-1)$ matrix, the first step of symmetric elimination is the factorization

$$A^{(0)} = \begin{bmatrix} \sqrt{d} & 0 \\ \frac{r}{\sqrt{d}} & I_{n-1} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \overline{H} - \frac{rr^T}{d} \end{bmatrix} \begin{bmatrix} \sqrt{d} & \frac{r^T}{\sqrt{d}} \\ 0 & I_{n-1} \end{bmatrix}. \quad (1)$$

Letting $A^{(1)} = \overline{H} - rr^T/d$, the factorization is then completed by recursively applying the basic step in (1) to $A^{(1)}$, $A^{(2)}$, and so on. Since A is sparse, some initially zero

0-8186-6555-6/94 $04.00 © 1994 IEEE

entries in A may become nonzero in L during factoring A into LL^T, which are called *fills* or *fill-ins*.

As observed by Rose, symmetric Gaussian elimination can be interpreted by a sequence of *elimination graphs*

$$G_A = G_{A^{(0)}} \rightarrow G_{A^{(1)}} \rightarrow \cdots \rightarrow G_{A^{(n-1)}} \rightarrow G_{A^{(n)}} = \varnothing.$$

where graph $G_{A^{(i)}}$ of $A^{(i)}$ is obtained from that of $A^{(i-1)}$ by (i) deleting v_i and its incident edges from $G_{A^{(i-1)}}$, and (ii) adding edges to $G_{A^{(i-1)}}$ so that nodes adjacent to v_i are pairwise adjacent in $G_{A^{(i)}}$.

For our purpose we confine the above graph model to a more specific one. Let G_F be the filled graph of G_A, where $F = L + L^T$ denotes the filled matrix of A, i.e.,

$$G_F = \bigcup_{i=0}^{n} G_{A^{(i)}}.$$

It is meaningful since the filled matrix of A has to be determined before the numerical factorization proceeds. So, we refer the factoring process of A as a sequence of elimination graphs of G_F such that from $G_{F^{(i-1)}}$ to $G_{F^{(i)}}$ only v_i and its incident edges are eliminated since all nodes adjacent to v_i has been pairwise connected.

2.2. Parallel sparse Cholesky factorizations

It is well-known that the algorithmic form of Cholesky factorization of A can be viewed as a triple nested loop containing the following statement [1]

$$a_{ij} \leftarrow a_{ij} - l_{ik}l_{jk}$$

where $L = (l_{ij})$ is the Cholesky factor.

Depending on which of the three indices is placed at the outmost loop there are three basic forms [4]: *column-Cholesky*, *row-Cholesky*, and *submatrix-Cholesky*. In some articles the column-Cholesky and submatrix-Cholesky are also known as *fan-in* and *fan-out* respectively. Recently, a sophisticated variant of the submatrix-Cholesky factorization, called *multifrontal* method [2], has been proposed and soon became a competitor to the other Cholesky factorizations. This paper is devoted to these four algorithms.

To exploit the potential parallelism existing in sparse matrix factorization, a commonly used structure is *elimination tree* [13]. Duff [3] has shown that the elimination tree also can be acted as an assembly tree for multifrontal method. An elimination tree T_A associated with the Cholesky factor L of matrix A is a tree containing the same nodes as the filled graph of A and for each v_k with $k < n$, its parent node is $v_p = parent(v_k)$, where $p = \min\{j \mid j > k \text{ and } l_{jk} \neq 0\}$. A related definition of *parent* is *child*, where $child(v_k) = \{v_c \mid parent(v_c) = v_k\}$. On the basis of the elimination tree model, we describe in Algorithms 1 to 4 respectively the parallel sparse column-, row-, submatrix-Cholesky and multifrontal methods, which are considered on a distributed-memory multiprocessor with the following assumptions:

(1) There is an unlimited number of processors that connected via a network of sufficiently wide bandwidth.

Algorithm 1. Column-Cholesky factorization

```
for $j$ := 1 to $n$ do in parallel
    while $v_j$ is not a leaf node do waiting;
    for $k$ := 1 to $j - 1$ and $l_{jk} \neq 0$ do
        for $i$ := $j$ to $n$ and $l_{ik} \neq 0$ do
            $a_{ij} \leftarrow a_{ij} - l_{ik}l_{jk}$;
    $l_{jj} \leftarrow \sqrt{a_{jj}}$;
    for $i$ := $j + 1$ to $n$ and $a_{ij} \neq 0$ do
        $l_{ij} \leftarrow a_{ij} / l_{jj}$;
    eliminate node $v_j$ from the elimination tree;
```

Algorithm 2. Row-Cholesky factorization.

```
for $i$ := 1 to $n$ do in parallel
    while $v_i$ is not a leaf node do waiting;
    for $k$ := 1 to $i - 1$ and $a_{ik} \neq 0$ do
        $l_{ik} \leftarrow a_{ik} / l_{kk}$;
        for $j$ := $k + 1$ to $i$ and $l_{jk} \neq 0$ do
            $a_{ij} \leftarrow a_{ij} - l_{ik}l_{jk}$;
    $l_{ii} \leftarrow \sqrt{a_{ii}}$;
    eliminate node $v_i$ from the elimination tree;
```

Algorithm 3. Submatrix-Cholesky factorization.

```
for $k$ := 1 to $n$ do in parallel
    while $v_k$ is not a leaf node do waiting;
    for $j$ := $k$ to $n$ and $a_{jk} \neq 0$ do
        $a_{jk} \leftarrow a_{jk} + \sum_{i<k \text{ and } l_{ki}\neq 0} t_{jk}^{(i)}$;
    $l_{kk} \leftarrow \sqrt{a_{kk}}$;
    for $j$ := $k + 1$ to $n$ and $a_{jk} \neq 0$ do
        $l_{jk} \leftarrow a_{jk} / l_{kk}$;
    for $j$ := $k + 1$ to $n$ and $l_{jk} \neq 0$ do
        for $i$ := $k + 1$ to $n$ and $l_{ik} \neq 0$ do
            $t_{ij}^{(k)} \leftarrow -l_{ik}l_{jk}$;
    eliminate node $v_k$ from the elimination tree;
```

Algorithm 4. Multifrontal method.

```
for $k$ := 1 to $n$ do in parallel
    while $v_k$ is not a leaf node do waiting;
    create frontal matrix $F^{(k)} \leftarrow A_{*k} + \sum_{v_c \in child(v_k)} \overline{F}^{(c)}$;
    apply a sequential dense Cholesky to factor the first
        column $F_{*1}^{(k)}$ of $F^{(k)}$;
    $L_{*k} \leftarrow F_{*1}^{(k)}$;
    $\overline{F}^{(k)} \leftarrow F^{(k)} - F_{*1}^{(k)}$;
    eliminate node $v_k$ from the elimination tree;
```

(2) The column-oriented distribution is used for column-, submatrix-Cholesky and multifrontal; row-oriented distribution is used for row-Cholesky. Each processor is solely responsible for the task of maintaining and updating its column or row.

(3) For simplicity, we ignore the underlying interconnecting and routing topology, and assume a message (a column or a row) can be transferred within a fixed time.

In order to keep the Cholesky methods consistent in generic form, the communication is assumed to be invoked by the computation without an explicit indication. The multifrontal method, however, is rather complicated, and so we give only an informal description; details can be found in the original paper by Duff and Reid [2]. In Algorithm 3, notation $t_{ij}^{(k)}$ denotes an intermediate updated factor that is generated in the factorization of column k and will be accumulated to entry a_{ij} when factoring column j, for $j > k$. In Algorithm 4, $F^{(k)}$ represents the frontal matrix associated with column k and $\overline{F}^{(k)}$ the remaining frontal matrix after the removal of the first column; A_{*k} and L_{*k} simply denote column k of A and L, respectively.

3. The evaluation criteria

3.1. Definition

In [9, 10] the authors have defined two evaluation criteria, *parallel computation time* and *parallel communication time*, to discuss the ordering problem, which indeed represent the computation and communication times needed to complete a Cholesky factorization in parallel. We adopt these two criteria and, for self-contained, we repeat the definitions in the following.

For v_i, $1 \le i \le n$, let $mtp(v_i)$ denote the number of multiplicative operations (square root, multiplication, and division) and $msg(v_i)$ the number of messages (columns or rows) acquired by v_i. The parallel computation time and communication time to complete node v_i are corresponding to the cost of a critical path from some leaf node to v_i, i.e.,

$$Comp(v_i) = \begin{cases} mtp(v_i), & \text{if } v_i \text{ is leaf} \\ mtp(v_i) + Mchild\_Comp(v_i), & \text{otherwise} \end{cases}$$

$$Comm(v_i) = \begin{cases} msg(v_i), & \text{if } v_i \text{ is leaf} \\ msg(v_i) + Mchild\_Comm(v_i), & \text{otherwise} \end{cases}$$

where

$Mchild\_Comp(v_i) = \max\{Comp(x) \mid x \in child(v_i)\}$,

$Mchild\_Comm(v_i) = \max\{Comm(x) \mid x \in child(v_i)\}$.

Then the parallel computation time and communication time required to complete the factorization, denoted as *Comp* and *Comm*, are equal to $Comp(v_n)$ and $Comm(v_n)$ respectively. Moreover, we introduce abbreviations, C, R, S and M as subscripts to distinguish different Cholesky factorizations, e.g., $mtp_S(v)$ represents the number of multiplicative operations associated with node v under submatrix-Cholesky factorization.

3.2. Formulation of the criteria

To study the behavior of and to distinguish the superiority among various Cholesky factorizations, we have to formulate the criteria.

Consider the filled graph G_F of matrix A. For each node v in G_F, we denote its adjacent set as $adj_{G_F}(v)$ and its degree as $deg_{G_F}(v)$, where $deg_{G_F}(v) = |adj_{G_F}(v)|$. The *prior* (*monotone*) adjacent set $Padj_{G_F}(v)$ ($Madj_{G_F}(v)$) is the set of all nodes adjacent to and numbered lower (higher) than v; $Pdeg_{G_F}(v)$ ($Mdeg_{G_F}(v)$) denote the size of $Padj_{G_F}(v)$ ($Madj_{G_F}(v)$). In what follows all discussions are related to G_F, we thus omit the subscript, G_F, in the above notations for simplicity.

A clique is a set of nodes with the property that all its members are pairwise adjacent; moreover, if no other node can be added while preserving the pairwise adjacent property, then the clique is called *maximal*. Assume G_F comprises $K_1, K_2, \ldots, K_q$ maximal cliques. A maximal clique K_j, $1 \le j \le q$, might change as the elimination proceeds and so similar to $G_{F^{(i-1)}}$ we denote the *residual clique* [8] of K_j after the ith elimination step as $K_j^{(i)}$. More precisely,

$$K_j^{(i)} = \begin{cases} K_j^{(i-1)} - \{v_i\}, & \text{if } v_i \in K_j^{(i-1)}, \\ K_j^{(i-1)}, & \text{otherwise.} \end{cases}$$

It should be noticed that a residual clique is not necessarily maximal. Furthermore, for node v_i (remember it is corresponding to column or row i of A and is eliminated in the ith step) we define $\Omega(v_i)$ as the identity of the maximal residual clique that contains v_i when v_i is eliminated. In other words, $K_{\Omega(v_i)}^{(i-1)}$ is the only maximal residual clique containing v_i in $G_{F^{(i-1)}}$, which indeed consists of the set of nodes adjacent to and ordered after v_i plus v_i, i.e., $K_{\Omega(v_i)}^{(i-1)} = Madj(v_i) \cup \{v_i\}$ and thus $|K_{\Omega(v_i)}^{(i-1)}| = Mdeg(v_i) + 1$. The notion $K_{\Omega(v_i)}^{(j)}$, for $1 \le j \le n$, then represents the residual clique of $K_{\Omega(v_i)}$.

The following equalities are useful for our derivations.

$$Padj(v_j) = \{v_k \mid 1 \le k \le j-1 \text{ and } l_{jk} \ne 0\} \qquad (2)$$

$$\sum_{j \le i \le n, l_{ik} \ne 0} 1 = \left|K_{\Omega(v_k)}^{(j-1)}\right|, \text{ for } k \le j-1 \qquad (3)$$

$$\left|K_{\Omega(v_i)}^{(i)}\right| = \left|K_{\Omega(v_i)}^{(i-1)}\right| - 1 \qquad (4)$$

Lemma 1. *The numbers of multiplicative operations associated with node v_j in T_A for column-, row-, submatrix-Cholesky, and multifrontal factorizations can be individually formulated as*

$$mtp_C(v_i) = \sum_{v_j \in Padj(v_i)\cup\{v_i\}} \left|K^{(i-1)}_{\Omega(v_j)}\right|,$$
$$mtp_R(v_i) = \sum_{v_j \in Padj(v_i)\cup\{v_i\}} \left(\left|K^{(j-1)}_{\Omega(v_j)}\right| - \left|K^{(i)}_{\Omega(v_j)}\right|\right),$$
$$mtp_S(v_i) = mtp_M(v_i) = \sum_{v_j \in Madj(v_i)\cup\{v_i\}} \left|K^{(j-1)}_{\Omega(v_i)}\right|.$$

Proof. We prove the case of $mtp_C(v_i)$ first. For each iteration of "for j" loop in Algorithm 1, there are $\sum_{1\le k\le j-1, l_{jk}\ne 0}\sum_{j\le i\le n, l_{ik}\ne 0} 1$ multiplications, one square root, and $\sum_{j+1\le i\le n, l_{ij}\ne 0} 1$ divisions. Totally, the number of multiplicative operations is:

$$\sum_{\substack{1\le k\le j-1\\ l_{jk}\ne 0}} \sum_{\substack{j\le i\le n\\ l_{ik}\ne 0}} 1 + 1 + \sum_{\substack{j+1\le i\le n\\ l_{ij}\ne 0}} 1$$
$$= \sum_{\substack{1\le k\le i-1\\ l_{jk}\ne 0}} \left|K^{(j-1)}_{\Omega(v_k)}\right| + 1 + \left|K^{(j)}_{\Omega(v_j)}\right|, \text{ by (3)}$$
$$= \sum_{v_k \in Padj(v_j)} \left|K^{(j-1)}_{\Omega(v_k)}\right| + \left|K^{(j-1)}_{\Omega(v_j)}\right|, \text{ by (2) and (4)}$$
$$= \sum_{v_k \in Padj(v_j)\cup\{v_j\}} \left|K^{(j-1)}_{\Omega(v_k)}\right|.$$

The lemma then follows by simple index substitution.

As for other cases, we can prove them by observing Algorithms 2, 3, 4, and similar derivations. □

For submatrix-Cholesky and multifrontal factorizations, their *mtp*'s are the same so we will denote that as $mtp_{SM}(v_i)$. Furthermore, be aware that $Madj(v_i) \cup \{v_i\} = K^{(i-1)}_{\Omega(v_i)}$ and thus

$$\sum_{v_j \in Madj(v_i)\cup\{v_i\}} \left|K^{(j-1)}_{\Omega(v_i)}\right| = \sum_{v_j \in K^{(i-1)}_{\Omega(v_i)}} \left|K^{(j-1)}_{\Omega(v_i)}\right|,$$

whose structure corresponds to a dense lower triangular matrix. Henceforth, we can rewrite the formula as

$$mtp_{SM}(v_i) = \tfrac{1}{2}\left|K^{(i-1)}_{\Omega(v_i)}\right|\left(\left|K^{(i-1)}_{\Omega(v_i)}\right|+1\right)$$
$$= \tfrac{1}{2}(Mdeg(v_i)+1)(Mdeg(v_i)+2).$$

Lemma 2. *The numbers of messages (columns or rows) acquired to update node v_i in T_A for column-, row-, submatrix-Cholesky, and multifrontal factorizations can be individually formulated as*

$$msg_C(v_i) = msg_R(v_i) = msg_S(v_i) = Pdeg(v_i),$$
$$msg_M(v_i) = \sum_{v_j \in child(v_i)} Mdeg(v_j).$$

Proof. The derivation is similar to Lemma 1 by observing the remote accessing data in Algorithms 1 to 4 and applying the corresponding graph notations. □

For column-, row-, and submatrix-Cholesky factorizations, their *msg*'s are the same so we will denote that as $msg_{CRS}(v_i)$.

4. Evaluation

4.1. Theoretical results

In this subsection, we derive the evaluation of the four Cholesky factorization algorithms. A useful property quoted from Schreiber [13] is stated first.

Lemma 3. (Schreiber [13]) *For each node v_j, $1 \le j < n$, $Madj(v_j)$ is a subset of nodes on the path from v_j to the root v_n of T_A.* □

Theorem 4. *On solving a given sparse matrix, the computation time of submatrix-Cholesky (multifrontal) factorization is no more than that of column-Cholesky factorization.*

Proof. This theorem can be proven if we can prove that for any path, say *Path*, from a leaf to the root on any elimination tree,

$$\sum_{v_j \in Path} mtp_C(v_j) \ge \sum_{v_i \in Path} mtp_{SM}(v_i).$$

By Lemmas 1, 3 and the definitions of *Madj* and *Padj*, we have the following derivations:

$$\sum_{v_j \in Path} mtp_C(v_j) = \sum_{v_j \in Path} \sum_{v_i \in Padj(v_j)\cup\{v_j\}} \left|K^{(j-1)}_{\Omega(v_i)}\right|$$
$$= \sum_{v_j \in Path} \sum_{\substack{v_i \in Path\\ \& v_i \in Padj(v_j)\cup\{v_j\}}} \left|K^{(j-1)}_{\Omega(v_i)}\right| + \sum_{v_j \in Path} \sum_{\substack{v_i \notin Path\\ \& v_i \in Padj(v_j)}} \left|K^{(j-1)}_{\Omega(v_i)}\right|$$
$$\ge \sum_{v_j \in Path} \sum_{\substack{v_i \in Path\\ \& v_i \in Padj(v_j)\cup\{v_j\}}} \left|K^{(j-1)}_{\Omega(v_i)}\right|$$
$$= \sum_{v_i \in Path} \sum_{\substack{v_j \in Path\\ \& v_j \in Madj(v_i)\cup\{v_i\}}} \left|K^{(j-1)}_{\Omega(v_i)}\right|$$
$$= \sum_{v_i \in Path} \sum_{v_j \in Madj(v_i)\cup\{v_i\}} \left|K^{(j-1)}_{\Omega(v_i)}\right| = \sum_{v_i \in Path} mtp_{SM}(v_i) \qquad \square$$

Theorem 5. *On solving a given sparse matrix, the computation time of submatrix-Cholesky (multifrontal) factorization is no more than that of row-Cholesky factorization.*

Proof. This theorem can be proven if we can prove that for any path, say *Path*, from a leaf to the root on any elimination tree,

$$\sum_{v_j \in Path} mtp_R(v_j) \ge \sum_{v_i \in Path} mtp_{SM}(v_i).$$

By Lemmas 1, 3 and the definitions of *Madj* and *Padj*, we have the following derivations:

$$\sum_{v_j \in Path} mtp_R(v_j)$$
$$= \sum_{v_j \in Path} \sum_{\substack{v_i \in Path\\ \& v_i \in Padj(v_j)\cup\{v_j\}}} \left(\left|K^{(i-1)}_{\Omega(v_i)}\right| - \left|K^{(j)}_{\Omega(v_i)}\right|\right)$$
$$+ \sum_{v_j \in Path} \sum_{\substack{v_i \notin Path\\ \& v_i \in Padj(v_j)\cup\{v_j\}}} \left(\left|K^{(i-1)}_{\Omega(v_i)}\right| - \left|K^{(j)}_{\Omega(v_i)}\right|\right)$$
$$\ge \sum_{v_j \in Path} \sum_{\substack{v_i \in Path\\ \& v_i \in Padj(v_j)\cup\{v_j\}}} \left(\left|K^{(i-1)}_{\Omega(v_i)}\right| - \left|K^{(j)}_{\Omega(v_i)}\right|\right)$$
$$= \sum_{v_i \in Path} \sum_{\substack{v_j \in Path\\ \& v_j \in Madj(v_i)\cup\{v_i\}}} \left(\left|K^{(i-1)}_{\Omega(v_i)}\right| - \left|K^{(j)}_{\Omega(v_i)}\right|\right)$$

$$= \sum_{v_i \in Path} \sum_{\substack{v_j \in Path \\ \& v_j \in Madj(v_i) \cup \{v_i\}}} \left| K_{\Omega(v_i)}^{(j-1)} \right| = \sum_{v_i \in Path} mtp_{SM}(v_i)$$

In derivations, note that, for each node v_i on the *Path*, $\sum_{v_j \in Path \& v_j \in Madj(v_i) \cup \{v_i\}} \left(\left| K_{\Omega(v_i)}^{(i-1)} \right| - \left| K_{\Omega(v_i)}^{(j)} \right| \right)$ is counted in decreasing order and $\sum_{v_j \in Path \& v_j \in Madj(v_i) \cup \{v_i\}} \left| K_{\Omega(v_i)}^{(j-1)} \right|$ is counted in increasing order, however their summations are the same. □

Theorem 6. *On solving a given sparse matrix, the computation time of column-Cholesky factorization is no more than that of row-Cholesky factorization.*

Proof. By observing derivations on the proofs of Theorems 4 and 5, this theorem will be true if we can prove that:

$$\sum_{v_j \in Path} \sum_{\substack{v_i \notin Path \\ \& v_i \in Padj(v_j)}} \left| K_{\Omega(v_i)}^{(j-1)} \right|$$

$$\leq \sum_{v_j \in Path} \sum_{\substack{v_i \notin Path \\ \& v_i \in Padj(v_j) \cup \{v_j\}}} \left(\left| K_{\Omega(v_i)}^{(i-1)} \right| - \left| K_{\Omega(v_i)}^{(j)} \right| \right)$$

By some careful derivations, we can prove the above inequality. □

It remains to evaluate the communication case.

Theorem 7. *On solving a given sparse matrix, the communication time of multifrontal method is no more than that of column-, row-, and submatrix-Cholesky factorization.*

Proof. This theorem can be proven if we can prove that for any path, say *Path*, from a leaf to the root on any elimination tree,

$$\sum_{v_j \in Path} msg_{CRS}(v_j) \geq \sum_{v_i \in Path} msg_M(v_i).$$

By Lemmas 2, 3, the definitions of *Madj*, *Padj*, *Mdeg*, *Pdeg*, and some properties of elimination tree, we have the following derivations:

$$\sum_{v_j \in Path} msg_{CRS}(v_j) = \sum_{v_j \in Path} \sum_{v_k \in Padj(v_j)} 1$$

$$= \sum_{v_j \in Path} \sum_{\substack{v_k \in Padj(v_j) \\ \& parent(v_k) \in Path}} 1 + \sum_{v_j \in Path} \sum_{\substack{v_k \in Padj(v_j) \\ \& parent(v_k) \notin Path}} 1$$

$$\geq \sum_{v_j \in Path} \sum_{\substack{v_k \in Padj(v_j) \\ \& parent(v_k) \in Path}} 1$$

$$= \sum_{parent(v_k) \in Path} \sum_{\substack{v_j \in Madj(v_k) \\ \& v_j \in Path}} 1$$

$$= \sum_{parent(v_k) \in Path} \sum_{v_j \in Madj(v_k)} 1$$

$$= \sum_{v_i \in Path} \sum_{v_k \in child(v_i)} \sum_{v_j \in Madj(v_k)} 1 = \sum_{v_i \in Path} msg_M(v_i) \qquad \square$$

As a brief summary of this subsection, we conclude:

(1) Submatrix-Cholesky (multifrontal) consumes the least parallel computation time, then the column-Cholesky and last the row-Cholesky.

(2) Multifrontal method suffers less parallel communication time than the other three methods, which suffer the same amount.

(3) To conclude, the multifrontal method is superior to the other three Cholesky factorization methods.

4.2. An example problem

In this subsection we illustrate the above evaluation with a simple problem whose Cholesky factor is shown in Figure 1. Figure 2 shows the (mtp_C, mtp_R, mtp_{SM}) and $[msg_{CRS}, msg_M]$ for each node of the corresponding elimination tree. A simple accumulating yields $Comp_C : Comp_R : Comp_{SM} = 48 : 62 : 30$ and $Comm_{CRS} : Comp_M = 19 : 15$.

$$\begin{pmatrix}
\times &&&&&&&& \\
& \times &&&&&&& \\
&& \times &&&&&& \\
&&& \times &&&&& \\
\times && \times && \times &&&& \\
& \times && \times && \times &&& \\
\times & \times &&& \times & \times & \times && \\
\times & \times & \times & \times & \times & \times & \times & \times & \\
&& \times & \times & \times & \times & \times & \times & \times
\end{pmatrix}$$

Figure 1. An example Cholesky factor.

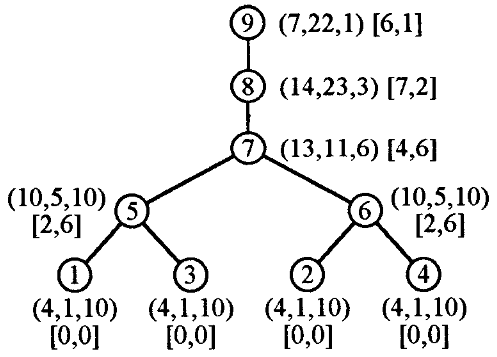

Figure 2. (mtp_C, mtp_R, mtp_{SM}) and $[msg_{CRS}, msg_M]$.

This example reveals that even for a simple problem a great variation exists in the performance of different Cholesky factorization methods. Since most real world problems have large sizes and sophisticated structures, we can expect a tremendous variation and we conjecture the variation may become infinite to the extreme.

Moreover, no matter which form of Cholesky factorization algorithms is used, the total number of multiplicative operations as well as the message transfer is the same. Thus the above example reveals an interesting but important phenomenon: the submatrix-Cholesky (and multifrontal) leads to the best load balancing in view of coarse-grained task scheduling, then the column-Cholesky, and last the row-Cholesky, which tells the reason of the theoretical results we obtained.

4.3. Experimental results

We make an experiment on a set of test matrices from the well-known Harwell-Boeing sparse matrices collection; the choice is following [8]. All matrices have been ordered by the minimum degree ordering [5] to reduce the number of fill-ins. The experimental results are reported in Table 1. It can be seen that the results have assented the theoretical evaluation in Section 4.1 and it seems there is a limitation on the ratio between different methods. But what the variation would be remains unknown and needs more theoretical study.

5. Conclusions

In this paper, we have defined, formalized the criteria and evaluated the performance of column-, row- and submatrix-Cholesky, and the multifrontal method. In the theoretical study, we have observed that the well load-balancing feature of multifrontal method makes it the supreme of the four methods.

However, the theoretical evaluation in this paper has some limitations. First, we only consider the coarse-grained task granularity (as exploited by elimination tree). In most practical implementations of parallel sparse Cholesky factorizations, they exploit the so called medium-grained granularity [11], and even fine-grained granularity [6]. Both cases are more complicated and deserved an advanced study. In addition, to reflect some subtle but influential overhead such as synchronization, indirect addressing and memory traffic, we also need a more sophisticated evaluation model.

References

[1] J. J. Dongarra, F. G. Gustavson and A. Karp, "Implementing linear algebra algorithms for dense matrices on a vector pipeline machine," *SIAM Review* 26, 1984, pp. 91–112.

[2] I. S. Duff and J. K. Reid, "The multifrontal solution of indefinite sparse symmetric linear equations," *ACM Trans. Math. Software* 9, 1983, pp. 302–325.

[3] I. S. Duff, "Parallel implementation of multifrontal schemes," *Parallel Comput.* 3, 1986, pp. 193–204.

[4] A. George, M. T. Heath and J. W. H. Liu, "Parallel Cholesky factorization on a shared-memory multiprocessor," *Lin. Alg. Appl.* 77, 1986, pp. 165–187.

[5] A. George and J. W. H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice Hall, Englewood Cliffs, NJ, 1981.

[6] J. R. Gilbert and R. Schreiber, "Highly parallel sparse Cholesky factorization," *SIAM J. Sci. Stat. Comput.* 13, 1992, pp. 1151–1172.

[7] M. T. Heath, E. Ng and B. W. Peyton, "Parallel algorithms for sparse linear systems," *SIAM Review* 33, 1991, pp. 420–460.

[8] J. G. Lewis, B. W. Peyton and A. Pothen, "A fast algorithm for reordering sparse matrices for parallel factorization," *SIAM J. Sci. Stat. Comput.* 10, 1989, pp. 1146–1173.

[9] W. Y. Lin and C. L. Chen, "Minimum completion time criterion for parallel sparse Cholesky factorization," in *Proc. International Conference on Parallel Processing*, Vol. III, St. Charles, IL, USA, 1993, pp. 107–114.

[10] W. Y. Lin and C. L. Chen, "Minimum communication time reordering for parallel sparse column-Cholesky factorization," in *Proc. International Conference on Parallel And Distributed Systems*, Taipei, Taiwan, 1993, pp. 147–151.

[11] J. W. H. Liu, "Computational models and task scheduling for parallel sparse Cholesky factorization," *Parallel Comput.* 3, 1986, pp. 327–342.

[12] D. J. Rose, "A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations," in R. C. Read, ed., *Graph Theory and Computing*, Academic Press, New York, 1972, pp. 183–217.

[13] R. Schreiber, "A new implementation of sparse Gaussian elimination," *ACM Trans. Math. Software* 8, 1982, pp. 256–276.

Table 1. Statistics of the test problems.

Key	Order	#nonzeros	$Comp_C$	$Comp_R$	$Comp_{SM}$	$Comm_{CRS}$	$Comm_M$
BCSPWR09	1723	2394	6089	8695	3616	1574	799
BCSPWR10	5300	8271	50562	72586	32066	6606	2879
BCSSTK08	1074	5943	784697	863997	620831	25373	16677
BCSSTK13	2003	40940	22051166	26897928	15430855	216448	128960
BCSSTM13	2003	9970	1850339	2069258	1283169	38433	24208
BLCKHOLE	2132	6370	762742	1022696	476237	27025	12630
CAN 1072	1072	5686	151533	205136	103412	9762	4992
DWT 2680	2680	11173	448620	584201	292997	29538	14719
LSHP3466	3466	10215	1156859	1546570	720903	41703	19512
GR 3030	900	4322	123050	172061	73056	8694	4039