ON EVALUATING ELIMINATION TREE BASED PARALLEL SPARSE CHOLESKY FACTORIZATIONS

WEN-YANG LIN[1] and CHUEN-LIANG CHEN[2]

[1] *Department of Information Management*
I-Shou University, Kaohsiung, Taiwan, ROC.
email: wylin@csa500.kpi.edu.tw

[2] *Department of Computer Science and Information Engineering*
National Taiwan University, Taipei, Taiwan, ROC.
email: clchen@csie.ntu.edu.tw

Abstract.

Though a variety of parallel sparse Cholesky factorizations have been developed and diverse experiments on various machines have been reported, it is still lack of theoretical evaluation because of the irregular structure of sparse matrices. This paper is an effort to such research. On the basis of elimination tree model, we compare the computation and communication perspective of four widely adopted parallel Cholesky factorization methods, including column-Cholesky, row-Cholesky, submatrix-Cholesky and multifrontal. The results show that the multifrontal method is superior to the others.

AMS subject classification: 05C50, 65F50, 65Y05.

Key words: Communication cost, distributed-memory multiprocessor, sparse matrix, parallel factorization, equivalent reordering, elimination tree.

1 Introduction.

In the direct solution of a sparse symmetric and positive definite linear system $Ax = b$, Cholesky factorization refers to the processing of decomposing 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 has emerged.

Though a variety of parallel sparse Cholesky factorizations have been proposed and are usually accompanied with some experimental results [5], [7], [13], [15], it is still lack of theoretical evaluation due to the irregularity of sparse matrices. The evaluation not only predicts the potential performance, but also exploits the possible limitation of a Cholesky factorization, and thus helps in choosing the appropriate method for a particular machine. This paper is an effort to such study. On the basis of elimination tree model, we compare the computation and communication persepctive of four widely adopted parallel 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, which is conformed with the current trend in the development of parallel Cholesky factorzations [13], [15].

The remaining of this paper is organized as follows. In Section 2, we review the graph notion used in matrix computation and the four Cholesky factorizations. In Section 3, we define the parallel completion time criterion, and formulate it in terms of graph notation. In Section 4, we evaluate and compare the performance of the four Cholesky factorizations. Experiments on some practical sparse matrices are also illustrated. 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 [14] and then confine it to a more specific graph model on which later derivations are based. Let $A = (a_{ij})$ be an $n \times n$ sparse symmetric positive definite matrix. The associated graph of A, $G_A = (V_A, E_A)$, can be constructed as below:

$$\begin{array}{rcl} V_A & = & \{v_i \mid v_i \text{corresponds to column/row} i \text{of} A, 1 \le i \le n\}, \\ E_A & = & \{(v_i, v_j) \mid a_{ij} \ne 0, 1 \le j < i \le n\}. \end{array}$$

The symmetric Gaussian elimination of A is described as follows. Letting

$$A = A^{(0)} = \left[\begin{array}{cc} d & r^T \\ r & \overline{H} \end{array} \right],$$

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

$$(2.1) \qquad A^{(0)} = \left[\begin{array}{cc} \sqrt{d} & 0 \\ \frac{r}{\sqrt{d}} & I_{n-1} \end{array} \right] \left[\begin{array}{cc} 1 & 0 \\ 0 & \overline{H} - \frac{rr^T}{d} \end{array} \right] \left[\begin{array}{cc} \sqrt{d} & r^T \\ 0 & I_{n-1} \end{array} \right].$$

Let $A^{(1)} = \overline{H} - rr^T/d$. The factorization is then completed by recursively applying the basic step in (2.1) to $A^{(1)}$, $A^{(2)}$, and so on. Since A is sparse, some initially zero entries in A may become nonzero in L during factoring A into LL^T, which are called *fill*s or *fill-in*s.

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

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

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

For our purpose we restrict 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.,

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

This description 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, and all nodes adjacent to v_i are pairwise connected.

2.2 Parallel sparse Cholesky factorization and computation model.

As apparent from the algorithmic form, a Cholesky factorization of A can be viewed as a triple nested loop containing the following statement [2]

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

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 [5] :*column-Cholesky*, *row-Cholesky* and *submatrix-Cholesky*. In the literature 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 [3], 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* [16]. Duff [4] showed 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 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 as well as unlimited number of memory modules connected via an interconnection network of sufficiently wide bandwidth.

(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.

In order to keep the Cholesky methods consistent in generic form, we assume the communication is invoked implicitly by the computation. The multifrontal method, however, is rather complicated, and so we give only an informal description. Readers should refer the original paper by Duff and Reid [3] for details. In Algorithm 3, notation $t_{ij}^{(k)}$ denotes an intermediate update that is generated in the factorization of column k and is accumulated to entry a_{ij} when factoring column j, for $j > k$. In Algorithm 4, $\mathcal{F}^{(k)}$ represents the frontal matrix associated with column k and $\overline{\mathcal{F}}^{(k)}$ 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.

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;
endfor
```

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;
endfor
```

Algorithm 3. *Submatrix-Cholesky factorization.*

```
for $k := 1$ to $n$ do in parallel
    while node $v_k$ is not a leaf node do waiting;
    for $j := k$ to $n$ and $t_{jk} \neq 0$ do
        $a_{jk} \leftarrow a_{jk} + \sum_{i<k \wedge l_{ki} \neq 0} t_{jk}^{(i)}$;
    $l_{kk} \leftarrow \sqrt{a_{kk}}$;
    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;
endfor
```

Algorithm 4. *Multifrontal method.*

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

3 The evaluation criterion.

3.1 Definitions.

On the basis of elimination tree model, we will evaluate the four Cholesky factorizations from two aspects: the computation spent and the communication overhead, including both message count and message volume. For this specific purpose, we adopt the time required to complete the whole factorization, the *parallel completion time*, as the basic criterion [11]. It, indeed, can be viewed as a critical path length of the elimination tree using the execution time (computation or communication) of each node as the cost.

For v_i, $1 \leq i \leq n$, let $mtp(v_i)$ denote the number of multiplicative operations (square root, multiplication, and division), $msc(v_i)$ the message (column or row) counts, and $msv(v_i)$ the message volumes acquired by v_i. The parallel computation and communication cost to complete node v_i can be formulated as

$$\begin{aligned}
Cp(v_i) &= \begin{cases} mtp(v_i), & \text{if} v_i \text{is leaf,} \\ mtp(v_i) + \widehat{Cp}(v_i), & \text{otherwise.} \end{cases} \\
Cmc(v_i) &= \begin{cases} msc(v_i), & \text{if } v_i \text{ is leaf,} \\ msc(v_i) + \widehat{Cmc}(v_i), & \text{otherwise.} \end{cases} \\
Cmv(v_i) &= \begin{cases} msv(v_i), & \text{if } v_i \text{ is leaf,} \\ msv(v_i) + \widehat{Cmv}(v_i), & \text{otherwise.} \end{cases}
\end{aligned}$$

where

$$\begin{aligned}
\widehat{Cp}(v_i) &= \max\{Cp(x) \mid x \in child(v_i)\}, \\
\widehat{Cmc}(v_i) &= \max\{Cmc(x) \mid x \in child(v_i)\}, \\
\widehat{Cmv}(v_i) &= \max\{Cmv(x) \mid x \in child(v_i)\}.
\end{aligned}$$

Then the parallel computation time and communication time required to complete the factorization, denoted as Cp, Cmc or Cmv, are equal to $Cp(v_n)$, $Cmc(v_n)$ or $Cmv(v_n)$, respectively. Moreover, we introduce subscripts C, R, S and M into the corresponding functions 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 node cost.

The parallel completion time criterion defined previously expresses exactly the cost spent on computation and communication. To study the behavior of and to distinguish the superiority among various Cholesky factorizations, we need a well-defined formulation of the cost spent on each node with respect to computation or communication. It has been shown well in the literature that the graphic notation is more concise and sensible. Thus, instead of presenting the formulations directly from the matrix notation, we will devise the corresponding graphic representation.

Consider the filled graph G_F of matrix A. For each node 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 a set of all nodes adjacent to and numbered lower (higher) than v; $Pdeg_{G_F}(v)$ ($Mdeg_{G_F}(v)$) denote the size of $Padj_{G_F}(v)$ ($Madj_{G_F}(v)$). Since all discussions hereafter are related to G_F, we omit the subscript G_F for simplicity.

A clique is a set of nodes with 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 of $K_1, K_2, ..., K_q$ maximal cliques. A maximal clique K_j, $1 \leq j \leq q$, might change as the elimination proceeds and so similar to $G_F^{(i-1)}$ we denote the *residual clique* [9] of K_j after the ith elimination step as $K_j^{(i)}$. More precisely,

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

Note that a residual clique is not necessarily maximal. For node v_i (remember it is corresponding to column/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 $i \leq j \leq n$, then represents the residual clique of $K_{\Omega(v_i)}$.

The following equalities are useful for our derivations.

$$Padj(v_j) = \{v_k \mid 1 \leq k \leq j-1 \text{ and } l_{jk} \neq 0\} \tag{3.1}$$

$$\sum_{j \leq i \leq n, l_{ik} \neq 0} 1 = \left|K_{\Omega(v_k)}^{(j-1)}\right|, \text{ for } k \leq j-1 \tag{3.2}$$

$$\left|K_{\Omega(v_i)}^{(i)}\right| = \left|K_{\Omega(v_i)}^{(i-1)}\right| - 1 \tag{3.3}$$

LEMMA 3.1. *The number of multiplicative operations associated with node v_i in T_A for column-, row-, submatrix-Cholesky, and multifrontal factorizations can be individually formulated as*

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

PROOF. We prove the case of $mtp_C(v_i)$ only; the other cases can be proved by similar derivations.

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

$$\begin{array}{rcl} \sum\limits_{\substack{1\le k\le j-1\\ l_{jk}\ne 0}}\ \sum\limits_{\substack{j\le i\le n\\ l_{ik}\ne 0}} 1 + 1 + \sum\limits_{\substack{j+1\le i\le n\\ l_{ij}\ne 0}} 1 & = & \sum\limits_{\substack{1\le k\le i-1\\ l_{jk}\ne 0}} \left|K^{(j-1)}_{\Omega(v_k)}\right| + 1 + \left|K^{(j)}_{\Omega(v_j)}\right|, \text{by (3.2)} \\ & = & \sum\limits_{v_k\in Padj(v_j)} \left|K^{(j-1)}_{\Omega(v_k)}\right| + \left|K^{(j-1)}_{\Omega(v_j)}\right|, \text{by (3.1, 3.3)} \\ & = & \sum\limits_{v_k\in Padj(v_j)\cup\{v_j\}} \left|K^{(j-1)}_{\Omega(v_k)}\right|. \end{array}$$

The lemma then follows by simple index substitutions. □

For submatrix-Cholesky and multifrontal factorizations, the involved computations are the same so we denote that as $mtp_{SM}(v_i)$. Furthermore, note that $Madj(v_i)\cup\{v_i\} = K^{(i-1)}_{\Omega(v_i)}$ and thus

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

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

$$\begin{array}{rcl} mtp_{SM}(v_i) & = & \frac{1}{2}\left|K^{(i-1)}_{\Omega(v_i)}\right|\left(\left|K^{(i-1)}_{\Omega(v_i)}\right| + 1\right) \\ & = & \frac{1}{2}(Mdeg(v_i)+1)(Mdeg(v_i)+2). \end{array}$$

LEMMA 3.2. *The number of messages (column or row), or message count, acquired to update node v_i in T_A for column-, row-, submatrix-Cholesky and* *multifrontal factorizations can be individually formulated as*

$$\begin{aligned} msc_C(v_i) &= msc_R(v_i) = msc_S(v_i) = Pdeg(v_i), \\ msc_M(v_i) &= \textstyle\sum_{v_j \in child(v_i)} Mdeg(v_j). \end{aligned}$$

PROOF. The derivation is similar to Lemma 3.1 by inspecting the remote accessing data in Algorithms 1 to 4 and applying the corresponding graph notation. □

For column-, row-, and submatrix-Cholesky factorizations, the message count acquired is the same so we denote that as $msc_{CRS}(v_i)$.

LEMMA 3.3. *The message volume acquired to update node v_i in T_A for column-, row-, submatrix-Cholesky and multifrontal factorizations can be individually formulated as*

$$\begin{aligned} msv_C(v_i) = msv_S(v_i) &= \textstyle\sum_{v_j \in Padj(v_i)} \left| K^{(i-1)}_{\Omega(v_j)} \right|, \\ msv_R(v_i) &= \textstyle\sum_{v_j \in Padj(v_i)} \left(\left| K^{(j-1)}_{\Omega(v_j)} \right| - \left| K^{(i-1)}_{\Omega(v_j)} \right| \right), \\ msv_M(v_i) &= \textstyle\sum_{v_j \in child(v_i)} \frac{1}{2} \left| K^{(i-1)}_{\Omega(v_j)} \right| \left(\left| K^{(i-1)}_{\Omega(v_j)} \right| + 1 \right). \end{aligned}$$

PROOF. The derivation is again similar to Lemma 3.1 by inspecting the remote accessing data in Algorithms 1 to 4 and applying the corresponding graph notation. □

Note in Lemma 3.3 the message volume of column- and submatrix-Cholesky factorizations is the same so we denote that as $msv_{CS}(v_i)$.

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 [16] is stated first.

LEMMA 4.1. *(Schreiber [16]) For each node v_j, $1 \leq j < n$. $Madj(v_j)$ is a subset of nodes on the path from v_j to the root v_n of T_A.*

THEOREM 4.2. *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 show that for any path, say ρ, from a leaf to the root on an elimination tree

$$\sum_{v_j \in \rho} mtp_C(v_j) \geq \sum_{v_i \in \rho} mtp_{SM}(v_i).$$

By Lemmas 3.1, 4.1 and the definitions of $Madj$ and $Padj$, we have the following derivations:

$$\begin{aligned}
\sum_{v_j \in \rho} mtp_C(v_j) &= \sum_{v_j \in \rho} \sum_{v_i \in Padj(v_j) \cup \{v_j\}} \left| K^{(j-1)}_{\Omega(v_i)} \right| \\
&= \sum_{v_j \in \rho} \sum_{\substack{v_i \in \rho \\ v_i \in Padj(v_j) \cup \{v_j\}}} \left| K^{(j-1)}_{\Omega(v_i)} \right| + \sum_{v_j \in \rho} \sum_{\substack{v_i \notin \rho \\ v_i \in Padj(v_j) \cup \{v_j\}}} \left| K^{(j-1)}_{\Omega(v_i)} \right| \\
&\geq \sum_{v_j \in \rho} \sum_{\substack{v_i \in \rho \\ v_i \in Padj(v_j)}} \left| K^{(j-1)}_{\Omega(v_i)} \right| \\
&= \sum_{v_i \in \rho} \sum_{\substack{v_j \in \rho \\ v_j \in Madj(v_i) \cup \{v_i\}}} \left| K^{(j-1)}_{\Omega(v_i)} \right| \\
&= \sum_{v_i \in \rho} \sum_{v_j \in Madj(v_i) \cup \{v_i\}} \left| K^{(j-1)}_{\Omega(v_i)} \right| = \sum_{v_i \in \rho} mtp_{SM}(v_i).
\end{aligned}$$

$\square$

The idea behind the above proof is that the execution time of v_i under submatrix-Cholesky would be dispersed to the execution times of its ancestors when solving via column-Cholesky; thus increases the total completion time on the path. Similar phenomena can be observed on other forms of factorization transfer.

THEOREM 4.3. *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 show that for any path, say ρ, from a leaf to the root on any elimination tree,

$$\sum_{v_j \in \rho} mtp_R(v_j) \geq \sum_{v_i \in \rho} mtp_{SM}(v_i).$$

By Lemmas 3.1, 4.1 and the definitions of $Madj$ and $Padj$, we have the following derivations:

$$\begin{aligned}
\sum_{v_j \in \rho} mtp_R(v_j) &= \sum_{v_j \in \rho} \sum_{\substack{v_i \in \rho \\ v_i \in Padj(v_j) \cup \{v_j\}}} \left(\left| K^{(i-1)}_{\Omega(v_i)} \right| - \left| K^{(j)}_{\Omega(v_i)} \right| \right) + \\
&\quad \sum_{v_j \in \rho} \sum_{\substack{v_i \notin \rho \\ v_i \in Padj(v_j)}} \left(\left| K^{(i-1)}_{\Omega(v_i)} \right| - \left| K^{(j)}_{\Omega(v_i)} \right| \right)
\end{aligned}$$

$$\begin{aligned}
&\geq \sum_{v_j \in \rho} \sum_{\substack{v_i \in \rho \\ v_i \in Padj(v_j) \cup \{v_j\}}} \left(\left| K^{(i-1)}_{\Omega(v_i)} \right| - \left| K^{(j)}_{\Omega(v_i)} \right| \right) \\
&= \sum_{v_i \in \rho} \sum_{\substack{v_j \in \rho \\ v_j \in Madj(v_i) \cup \{v_i\}}} \left(\left| K^{(i-1)}_{\Omega(v_i)} \right| - \left| K^{(j)}_{\Omega(v_i)} \right| \right) \\
&= \sum_{v_i \in \rho} \sum_{v_j \in Madj(v_i) \cup \{v_i\}} \left| K^{(j-1)}_{\Omega(v_i)} \right| = \sum_{v_i \in \rho} mtp_{SM}(v_i).
\end{aligned}$$

□

Note that in the last step of the above derivations, the summation

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

is counted from 1 to $Mdeg(v_j) + 1$ while

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

is counted from $Mdeg(v_j) + 1$ to 1; both yield the same result.

Theorem 4.4. *On solving a given sparse matrix, the computation time of column-Cholesky factorization is no more than that of row-Cholesky factorization.*

Proof. Similar to the proofs in Theorems 4.2 and 4.3, this theorem will be true if we can show that

$$\sum_{v_j \in \rho} \sum_{\substack{v_i \notin \rho \\ v_i \in Padj(v_j)}} \left| K^{(j-1)}_{\Omega(v_i)} \right| \leq \sum_{v_j \in \rho} \sum_{\substack{v_i \notin \rho \\ v_i \in Padj(v_j)}} \left(\left| K^{(i-1)}_{\Omega(v_i)} \right| - \left| K^{(j)}_{\Omega(v_i)} \right| \right).$$

The above inequality follows by some careful derivations, . □

Now it remains to evaluate the communication case.

Theorem 4.5. *On solving a given sparse matrix, the communication cost of multifrontal method is no more than that of column-, row-, and submatrix-Cholesky factorization.*

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

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

By Lemmas 3.2, 4.1, the definitions of $Madj$, $Padj$, $Mdeg$, $Pdeg$ and some properties of elimination tree, we have the following derivations:

$$\begin{aligned}
\sum_{v_j \in \rho} msg_{CRS}(v_j) &= \sum_{v_j \in \rho} \sum_{v_k \in Padj(v_j)} 1 \\
&= \sum_{v_j \in \rho} \sum_{\substack{v_k \in Padj(v_j) \\ parent(v_k) \in \rho}} 1 + \sum_{v_j \in \rho} \sum_{\substack{v_k \in Padj(v_j) \\ parent(v_k) \notin \rho}} 1 \\
&\geq \sum_{v_j \in \rho} \sum_{\substack{v_k \in Padj(v_j) \\ parent(v_k) \in \rho}} 1 \\
&= \sum_{parent(v_k) \in \rho} \sum_{\substack{v_j \in Madj(v_k) \\ v_j \in \rho}} 1 \\
&= \sum_{parent(v_k) \in \rho} \sum_{v_j \in Madj(v_k)} 1 \\
&= \sum_{v_i \in \rho} \sum_{v_k \in child(v_i)} \sum_{v_j \in Madj(v_k)} 1 = \sum_{v_i \in \rho} msg_M(v_i).
\end{aligned}$$

$\square$

For the similarity between the functions for message volume and the computation counterparts, the following theorems are stated without proving.

THEOREM 4.6. *On solving a given sparse matrix, the message volume of multifrontal method is no more than that of column- and submatrix-Cholesky factorization.*

THEOREM 4.7. *On solving a given sparse matrix, the message volume of column- and submatrix-Cholesky factorizations is no more than that of row-Cholesky factorization.*

THEOREM 4.8. *On solving a given sparse matrix, the message volume of multifrontal method is no more than that of row-Cholesky factorization.*

We conclude the following as a summary of this subsection:

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

(2) Multifrontal method suffers less message count than the other three methods, which suffer the same amount.

(3) Multifrontal method suffers the least message volume, then the column- and submatrix-Cholesky, and last the row-Cholesky.

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

4.2 An example problem.

We illustrate the above evaluation with a simple problem whose Cholesky factor is shown in Figure 4.1. Figure 4.2 shows the $(mtp_C,\ mtp_R,\ mtp_{SM})$, $[msc_{CRS},\ msc_M]$ and $\{msv_{CS}, msv_R, msv_M\}$ for each node of the corresponding elimination tree. A simple accumulating yields $Cp_C : Cp_R : Cp_{SM} = 48 : 62 : 30$, $Cmc_{CRS} : Cmc_M = 19 : 15$ and $Cmv_{CS} : Cmv_R : Cmv_M = 34 : 38 : 28$.

This example reveals that even a simple problem would yield a great variation in the performance of different Cholesky factoring methods. Since most real problems have large sizes and sophisticated structures, we expect a tremendous variation and conjecture that may be infinite to the extreme.

Moreover, no matter which form of Cholesky factorizations is used, the total number of multiplicative operations as well as the message transfer is the same. It seems to reveal that the multifrontal (and submatrix-Cholesky) leads to the best load-balancing, then the column-Cholesky, and last the row-Cholesky. But research has shown that column-Cholesky usually achieves better load-balancing than the other methods [1],[8]. The clue to the superiority of submatrix-Cholesky lies in the granularity of tasks in each level of the elimination tree. For submatrix-Cholesky, the node cost tends to decrease toward the root of the elimination tree, which is propotional to the degree of parallelism in each tree level. This implies more computations can be performed in parallel when we have more processors activated; thus shortens the completion time to finish the factorization.

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 follows the study in [9]. All matrices were ordered by the minimum degree ordering [6] to reduce the number of fill-ins and followed by the Jess and Kees algorithm [10] to enrich the parallelism. The critical cost of each matrix with respect to computation or communication under different Cholesky factorization methods is reported in Table 4.1 and Table 4.2. The result apparently assents to the theoretical evaluation in Section 4.1 and there seems to be a limitation on the ratio between different methods. But whether it is true and what the variation would be remains unknown and needs further study.

5 Conclusions.

In this paper, we have utilized the parallel completion time criterion to evaluate 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 makes it the supreme of the four methods.

However, the 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 [12], and even fine-grained granularity [7]. 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. C. Ashcraft, *The fan-both family of column-based distributed Cholesky factorization algorithms*, in A. George, ed., *Graph Theory and Sparse Matrix Computation*, The IMA Volumns in Mathematics and its Applications, Vol. 56, Springer-Verlag, New York, 1993, 159–190.
2. 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), 99–112.
3. I. S. Duff and J. K. Reid, *The multifrontal solution of indefinite sparse symmetric linear equations*, ACM Trans. Math. Software, 9 (1983), 302–325.
4. I. S. Duff, *Parallel implementation of multifrontal schemes*, Parallel Computing, 3 (1986), 193–204.
5. A. George, M.T. Heath and J.W.-H. Liu, *Parallel Cholesky factorization on a shared-memory multiprocessor*, Lin. Alg. Appl., 77 (1986), 165–187.
6. A. George and J.W.-H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice Hall, Englewood Cliffs, NJ, 1981.
7. J. R. Gilbert and R. Schreiber, *Highly parallel sparse Cholesky factorization*, SIAM J. Sci. Stat. Comput., 13 (1992), 1151–1172.
8. M. T. Heath, E. Ng and B. W. Peyton, *Parallel algorithms for sparse linear systems*, SIAM Review, 33 (1991), 420–460.
9. 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), 1146–1173.
10. J. A. G. Jess and H. G. M. Kees, *A data structure for parallel L/U decomposition*, IEEE Trans. Comput., C-31 (1982), 231–239.
11. W. -Y. Lin and C. -L. Chen, *Minimum completion time criterion for parallel sparse Cholesky factorization*, in Proc. International Conference on Parallel Processing, III (1993), St. Charles, IL, USA, 107–114.

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

Figure 5.1: An example Cholesky factor.

12. J. W. -H. Liu, *Computational models and task scheduling for parallel sparse Cholesky factorization*, Parallel Computing, 3 (1986), 327–342.

13. R. Gupta and V. Kumar, *Optimally scalable parallel sparse Cholesky factorization*, in Proc. the Seventh SIAM Conference on Parallel Processing for Scientific Computing (1995), SIAM, 442–447.

14. 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, 183–217.

15. E. Rothberg, *Performance of panel and block approaches to sparse Cholesky factorization on the iPSC/860 and Paragon multicomputers*, SIAM J. Sci. Comput., 17 (1996), 699–713.

16. R. Schreiber, *A new implementation of sparse Gaussian elimination*, ACM Trans. Math. Software, 8 (1982), 256–276.

17. R. Schreiber, *Scalability of sparse direct solvers*, in A. George, ed., *Graph Theory and Sparse Matrix Computation*, The IMA Volumns in Mathematics and its Applications, Vol. 56, Springer-Verlag, New York, 1993, 191–209.

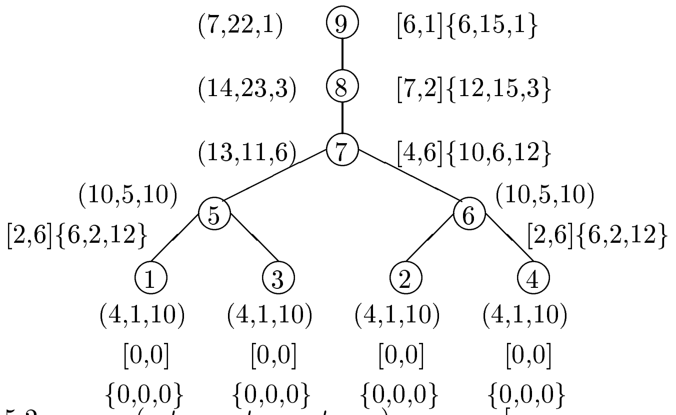

Figure 5.2: (mtp_C, mtp_R, mtp_{SM}) , $[msc_{CRS}, msc_M]$, and $\{msv_{CS}, msv_R, msv_M\}$.

Table 5.1: Computation statistics of the test matrices.

Key	Order	$\tau(A)$	Cp_C	Cp_R	Cp_{SM}
BCSPWR09	1723	2394	3664	5817	2140
BCSPWR10	5300	8271	43971	62156	28632
BCSSTK08	1074	5943	611554	768217	439262
BCSSTK13	2003	40940	3970126	7169318	8366244
BCSSTM13	2003	9970	1801953	2012053	1335285
BLCKHOLE	2132	6370	762742	1022696	476237
CAN 1072	1072	5686	151533	205136	103412
DWT 2680	2680	11173	363680	448920	261665
LSHP3466	3466	10215	921041	1332781	573938
GR 3030	900	4322	106525	157406	62549

Table 5.2: Communication statistics of the test matrices.

Key	Order	$\tau(A)$	Cmc_{CRS}	Cmc_M	Cmv_{CS}	Cmv_R	Cmv_M
BCSPWR09	1723	2394	1023	482	3284	4787	2166
BCSPWR10	5300	8271	5152	2303	41818	56903	28540
BCSSTK08	1074	5943	21556	12315	599788	746463	443033
BCSSTK13	2003	40940	215812	140111	3830847	6952866	8378914
BCSSTM13	2003	9970	35912	23078	1779168	1975898	1336984
BLCKHOLE	2132	6370	27025	12630	750321	995493	475183
CAN 1072	1072	5686	9762	4992	146770	195241	103186
DWT 2680	2680	11173	21231	11257	352844	427431	261582
LSHP3466	3466	10215	34608	15453	905871	1297879	571038
GR 3030	900	4322	7830	3469	103165	149459	62320