

On Optimal Fill-Preserving Orderings of Sparse Matrices for Parallel Cholesky Factorizations

Wen-Yang Lin

Dept. of Inform. Management
I-Shou University
Kaohsiung, Taiwan, ROC
wylin@csa500.isu.edu.tw

Chuen-Liang Chen

Dept. of Comput. Science and Inform. Engineering
National Taiwan University
Taipei, Taiwan, ROC
clchen@csie.ntu.edu.tw

Abstract

In this paper, we consider the problem of finding fill-preserving ordering of a sparse symmetric and positive definite matrix such that the reordered matrix is suitable for parallel factorization. We extended the unit-cost fill-preserving ordering into a generalized class that can adopt various aspects in parallel factorization, such as computation, communication and algorithmic diversity. Based on the elimination tree model, we show that as long as the node cost function for factoring a column/row satisfies two mandatory properties, we can deploy a greedy-based algorithm to find the corresponding optimal ordering. The complexity of our algorithm is $O(q \log q + \kappa)$, where q denote the number of maximal cliques, and κ the sum of all maximal clique sizes in the filled graph. Our experiments reveal that on the average, our minimum completion cost ordering (MinCP) would reduce up to 17% the cost to factor than minimum height ordering (Jess-Kees).

1 Introduction

In this paper, we consider the problem of finding fill-preserving sparse matrix orders for parallel factorization, which arises during the exploitation of parallelism in the direct solution of large sparse symmetric positive definite systems. Given a large sparse symmetric and positive definite matrix A , we want to determine an ordering which is appropriate in terms of preserving the sparsity and minimizing the cost to perform its Cholesky factorization in parallel.

Jess and Kees [8] was known as the first to propose a fill-preserving ordering strategy for parallel factorization. Their method has the desirable property of minimizing the height of the elimination tree among

the class of fill-preserving orderings. Some effective implementations of Jess and Kees algorithm have been proposed [9][15]. In [14], Joseph Liu used a tree rotation heuristic to find orderings, though not optimal, could substantially reduce the height of the elimination tree. Aspvall and Heggernes [1] also proposed an efficient algorithm to generate orderings for a minimum height elimination tree. But their work was limited to interval graphs, a subclass of chordal graphs.

Thus far, all approaches to this problem have been devoted to minimizing the height of the elimination tree. Such criterion is based on the assumption of unit cost for node elimination, though simple in nature, is improper for exhibiting various factors that affect the cost of parallel factorization, such as communication overhead and the algorithmic diversity of Cholesky form. In [10][11], we have produced some examples to illustrate how the cost discrepancy may grow infinitely. We have also solved the problem that specifically on adopting the operation count or communication number for factoring a column/row to generate a parallel pivoting sequence with the property that minimizes the critical completion cost for parallel factorization.

In this study, we extended the criterion of the elimination tree height to a more general class that is categorized as the nodal cost function satisfying two properties, called *independent* and *conservative* properties. In terms of such a class of ordering criteria we propose a greedy algorithm to find the optimal orderings for parallel factorization.

The remainder of this paper is organized as follows. In Section 2, we provide background material and introduce related notation. In Section 3, we first introduce the elimination tree based computation model on which all discussions are based. We then define the independent and conservative properties, and specify the generalized class of ordering criteria. Section 4 de-

scribes our greedy algorithm for finding optimal fill-preserving ordering. The proof of optimality and implementation details are also discussed there. Section 5 presents the experimental results on the Harwell-Boeing test set [4]. Finally, the conclusion and future work is summarized in Section 6.

2. Background

2.1. Sparse Cholesky factorization

Consider a system of linear equations $Ax = b$, where A is an $n \times n$ symmetric positive definite matrix, b is a given vector, and x is the unknown vector to be solved. In the direct solution of such linear systems, A is usually first decomposed into LL^T , which is known as Cholesky factorization, where L is lower triangular. Then the solution vector x is computed by solving two triangular systems $Ly = b$ and $L^T x = y$. When A is sparse, some initially zero entries in A may become nonzero in L , which are called *fill* or *fillin*. In order to reduce time and storage requirements, only the nonzero positions of L are stored and operated on during sparse Cholesky factorization. Hence, an ordering stage is typically applied first to reduce the fill in the factor L [5].

By permuting the three nested loops comprised of the single statement $a_{ij} \leftarrow a_{ij} - l_{ik}l_{jk}$, there are six algorithmic forms of Cholesky factorization. Depending on which of the three indices is placed in the outer most loop they can be classified as three basic forms: *column-Cholesky*, *row-Cholesky* and *submatrix-Cholesky*. Here, we consider the parallel versions of these three sparse Cholesky factorizations and a complicated derivation of sparse submatrix-Cholesky, multifrontal [3].

2.2. Graph notations

The nonzero structure of matrix A can be conveniently represented by an undirected graph $G = (V, E)$. The n nodes, v_1, v_2, \dots, v_n , correspond to the n columns/rows of the matrix, and an edge connect v_i and v_j if and only if the corresponding entry a_{ij} is nonzero. A graph G is called an "ordered graph" if a bijection $\alpha : \{1, 2, \dots, n\} \rightarrow V$ is defined. The bijection α is called the "ordering" of G . To simplify the discussion, we assume that the sparse matrix A has been ordered by a fill-reducing ordering, such as the minimum degree or nested dissection ordering.

For a node v in G , we denote its adjacent set as $adj(v, G)$ and its degree as $deg(v, G)$. A prior (monotone) adjacent set $Padj(v, G)$ ($Madj(v, G)$) is the set

of all nodes adjacent to and numbered lower (higher) than v , and $Pdeg(v, G)$ ($Mdeg(v, G)$) is the size of $Padj(v, G)$ ($Madj(v, G)$).

The structural effect of Gaussian elimination on the matrix can be easily modeled by a sequence of elimination graphs [5]: Initially, let $G^{(0)} = G$. In the i th step, for $1 \leq i \leq n$, a node v along with its incident edges are eliminated from $G^{(i-1)}$ and its adjacent set is mutually connected. The graph structure of the resulting Cholesky factor L of A , G^* , is called the *filled graph* of A and $G^* = \bigcup_{i=0}^{n-1} G^{(i)}$.

A clique is a set of nodes with the property that all of its members are pairwise adjacent. If no other node can be added while preserving the pairwise adjacent property, the clique is called *maximal*. It has been shown that a filled graph G^* can be represented as a set of maximal cliques. Assume that G^* comprises K_1, K_2, \dots, K_q maximal cliques. We define the *residual clique* of K_j after i elimination steps as $K_j^{(i)}$.

A node v in graph G is *simplicial* if $adj(v, G)$ is a clique. Two nodes, u and v , are *indistinguishable* if $\{u\} \cup adj(u, G) = \{v\} \cup adj(v, G)$. Two nodes are *independent* if there is no edge between them.

A *fill-preserving ordering* of A is a permutation such that the reordered matrix will suffer the same fill and require the same number of arithmetic operations for factorization. From the work of Rose [16], a filled graph is a chordal graph and always has at least one perfect ordering, i.e. an ordering with no fill. Therefore, finding a fill-preserving ordering for the fill-reducing ordered matrix A is equivalent to finding a perfect ordering of the filled graph G^* .

The elimination of a simplicial node has been shown that creates no filled edge. A perfect ordering thus can be obtained by finding simplicial nodes in the reduced graphs during the elimination process.

3 Computation model and cost specification

3.1 The elimination tree model

The *elimination tree* model [8] has been shown to be appropriate in exploiting the parallelism that exists in sparse matrix factorization. An elimination tree $T(A)$ of matrix A is a tree containing the same node set as G , and has an edge between two nodes v_i and v_j if $v_j = parent(v_i)$, where $parent(v_i) = \min \{v_k \mid k > i \text{ and } l_{ki} \neq 0\}$. Liu [13] has illustrated that the elimination tree model is appropriate for the three versions of Cholesky factorizations. Duff [3] also used the elimination tree as a structure in their multifrontal factorization.

Let $\Phi(v, G^*)$ specify the cost of eliminating a node v in the filled graph G^* with respect to the Cholesky factorization. It is natural to define the cost (called *completion cost*) to complete the parallel elimination at node v as the critical weighted path on the subtree root at node v , $T[v]$. More precisely, let CP denote the completion cost function. We have the following recursive definition

$$CP(v, G^*) = \begin{cases} \Phi(v, G^*), & v \text{ is a leaf,} \\ \Phi(v, G^*) + \max_{u \in \text{child}(v)} CP(u, G^*), & \text{else.} \end{cases}$$

The total completion cost, CP , to eliminate the entire graph or factorize the corresponding matrix is then equal to $CP(v_n, G^*)$. For clarity, we will use $CP(v)$ and $\Phi(v)$ instead of $CP(v_n, G^*)$ and $\Phi(v, G^*)$, and $CP_\alpha(v)$ and $\Phi_\alpha(v)$ for $CP(v, G_\alpha^*)$ and $\Phi(v, G_\alpha^*)$ when the ordering has to be distinguished.

3.2. Independent and conservative properties

Let ρ be a perfect ordering on G^* : $\rho : v_1, v_2, \dots, v_n$, such that the j -th and $j+1$ -st nodes, v_j and v_{j+1} , are both simplicial in $G_\rho^{*(j-1)}$. Then the ordering $\tilde{\rho} : v_1, v_2, \dots, v_{j-1}, v_{j+1}, v_j, v_{j+2}, \dots, v_n$ that differs from ρ only in the j -th and $j+1$ -st positions is also a perfect ordering. We define a class Ω of cost functions that satisfies the following independent and conservative properties. That is, $\Omega = \{\Phi : \Phi \text{ satisfies the independent and conservative properties}\}$.

DEFINITION 3.1. *Let v_j and v_{j+1} belong to two different simplicial cliques in $G_\rho^{*(j-1)}$. The cost function satisfies the independent property, if the relative order between v_j and v_{j+1} is irrelevant. That is,*

$$\Phi_{\tilde{\rho}}(v_{j+1}) = \Phi_\rho(v_{j+1}) \text{ and } \Phi_{\tilde{\rho}}(v_j) = \Phi_\rho(v_j).$$

DEFINITION 3.2. *Let v_j and v_{j+1} belong to the same simplicial clique in $G_\rho^{*(j-1)}$. The cost function satisfies the conservative property, if the sum of the cost of v_j and v_{j+1} is unchanged. That is,*

$$\Phi_\rho(v_j) + \Phi_\rho(v_{j+1}) = \Phi_{\tilde{\rho}}(v_j) + \Phi_{\tilde{\rho}}(v_{j+1}).$$

3.3. Example cost functions in Ω

There are various factors at affecting the specification of the cost function for eliminating nodes in parallel. On the basis of elimination tree model, we consider the parallel sparse column-, row-, submatrix-Cholesky and multifrontal methods on a distributed-memory multiprocessor with the following assump-

tions: 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.

It is difficult to define one cost function which is generally good for most cases. Instead, we choose to specify a cost function with respect to each circumstance. As an example, we consider the typical factors, computation and communication, with respect to different algorithmic forms of Cholesky factorization. Table 1 summarizes this category. The specification of each cost functions is described in Table 2, where the notation $K_{v_j}^{(i)}$ denote the residual set of the maximal clique K_{v_j} where a node v_j become simplicial. The derivation of these specifications is omitted here. We only show in Figure 1 the case for Φ_2 ; the details can be found in [10][11].

Table 1. A category of node cost functions.

		Col	Row	Sub	Mtf
comp	unit	Φ_1	Φ_1	Φ_1	Φ_1
	mult	Φ_2	Φ_3	Φ_4	Φ_4
comm	num	Φ_5	Φ_5	Φ_5	Φ_6
	vol	Φ_7	Φ_8	Φ_7	Φ_9

Table 2. Specification of the cost functions in Table 1.

Φ_1	1			
Φ_2	$\sum_{v_j \in \text{Padj}(v_i) \cup \{v_i\}} K_{v_j}^{(i-1)} $			
Φ_3	$\sum_{v_j \in \text{Padj}(v_i) \cup \{v_i\}} \left(K_{v_j}^{(j-1)} - K_{v_j}^{(i)} \right)$			
Φ_4	$\sum_{v_j \in \text{Madj}(v_i) \cup \{v_i\}} K_{v_i}^{(j-1)} $			
Φ_5	$\text{Pdeg}(v_i)$			
Φ_6	$\sum_{v_j \in \text{child}(v_i)} \text{Mdeg}(v_j)$			
Φ_7	$\sum_{v_j \in \text{Padj}(v_i)} K_{v_j}^{(i-1)} $			
Φ_8	$\sum_{v_j \in \text{Padj}(v_i)} \left(K_{v_j}^{(j-1)} - K_{v_j}^{(i-1)} \right)$			
Φ_9	$\sum_{v_j \in \text{child}(v_i)} \frac{1}{2} K_{v_j}^{(i-1)} $		$\left(K_{v_j}^{(i-1)} + 1 \right)$	

Note that function Φ_1 is the case for unit cost. That is, the criterion of using elimination tree height

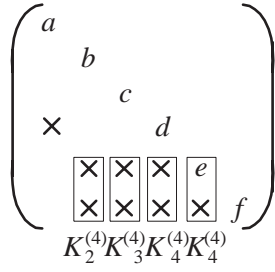


Figure 1. Illustration of nodal cost Φ_2 .

is merely a special case of the proposed class. Each function can be verified that satisfies the independent and conservative properties. As an illustration, consider the filled graph in Figure 1 again. The pictures in Figures 2 and 3 show the establishing of the independent and conservative properties for nodal cost function Φ_2 , which measures the multiplicative operations for column-Cholesky factorization.

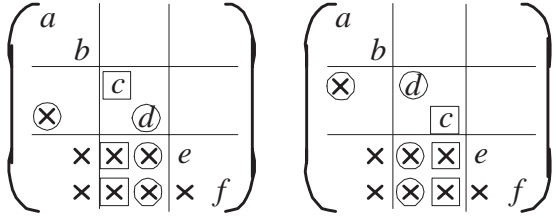


Figure 2. Illustration of the cost independent property.

4. A generic ordering algorithm

4.1. General description

Assume that the sparse matrix A has been ordered by a fill-reducing ordering and the filled graph be G^* . Our intention is to find a perfect ordering $\sigma : \{1, 2, \dots, n\} \rightarrow V$ on G^* to minimize the total completion cost.

The relation between perfect ordering and simplicial nodes gives us an intuition—use the elimination process model. In each elimination step, the nodes that are simplicial are identified. Then based on a greedy

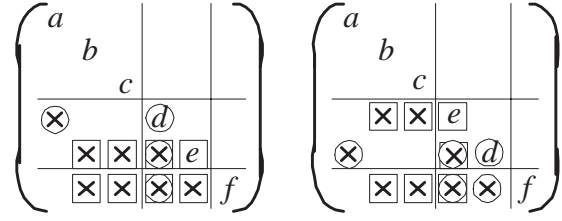


Figure 3. Illustration of the cost conservative property.

paradigm, one of the simplicial nodes having the minimum completion cost as defined in Section 3 is chosen to be labeled and eliminated next. This process is continuing until all nodes are eliminated. The algorithm is summarized as follow.

ALGORITHM. *Minimum completion time ordering algorithm. (MinCP)*

```

 $G^{(0)} \leftarrow G^*$ ;
for  $i = 1$  to  $n$  do
     $S \leftarrow$  the set of all simplicial nodes in  $G^{(i)}$ ;
    pick a simplicial node  $v$  with  $\min_{v \in S} CP(v)$ ;
     $\sigma^{(i)} \leftarrow v$ ;
     $G^{(i)} \leftarrow G^{(i-1)} \setminus \{v\}$ ;
endfor

```

The effectiveness of the above process lies in the selection of the next eliminated simplicial node. To make a choice, we have to evaluate the completion cost of each simplicial node. This can only be reached, however, when the simplicial node is actually be eliminated. In other words, it means that we have to determine the relative order between those simplicial nodes in the current elimination graph. We consider two cases: the relative order between independent simplicial nodes and that between indistinguishable simplicial nodes.

More specifically, let $\rho : v_1, v_2, \dots, v_n$ be a perfect ordering in which the j -th and $j + 1$ -st nodes, v_j and v_{j+1} are both simplicial in $G^{*(j-1)}$. Then a reordering of ρ which differs from only in the j -th and $j + 1$ -st positions is also a perfect ordering, i.e., $\tilde{\rho} : v_1, v_2, \dots, v_{j-1}, v_{j+1}, v_j, v_{j+2}, \dots, v_n$. We will deliberate, through conditions on Φ , the effect the interchange of v_j and v_{j+1} has on the completion cost, and then extend to the general case for a set of simplicial nodes $v_{j_1}, v_{j_2}, \dots, v_{j_s}$ in $G^{*(j-1)}$. For length limitation,

all properties are given without proof.

LEMMA 4.1. *If v_j and v_{j+1} are both simplicial and belong to different maximal cliques in $G^{*(j-1)}$, then $CP_\rho \sim = CP_\rho$.*

COROLLARY 4.2. *Assume that $v_j, v_{j+1}, \dots, v_{j+s}$ is an independent set of simplicial nodes in $G^{*(j-1)}$. Then, the relative order between $v_j, v_{j+1}, \dots, v_{j+s}$ is irrelevant with respect to the total completion cost.*

We next consider the case for indistinguishable nodes. To facilitate the discussion and simplify the notation, we introduce the *descendant cost* of a node v in $G^{*(j)}$, $1 \leq j \leq n$, as

$$D_j(v) = \max\{CP(u) \mid u \in \text{adj}(v, G^*) \cap \{v_1, v_2, \dots, v_j\}\}.$$

We don't refer to the children of v because this set is not determined until v is eliminated.

LEMMA 4.3. *If v_j and v_{j+1} are both simplicial and belonging to the same maximal clique in $G^{*(j-1)}$, and $D_{j-1}(v_j) \geq D_{j-1}(v_{j+1})$, then $CP_\rho \leq CP_\rho$.*

COROLLARY 4.4. *Assume that $v_j, v_{j+1}, \dots, v_{j+s}$ is an indistinguishable set of simplicial nodes in $G^{*(j-1)}$ and $D_{j-1}(v_j) \leq D_{j-1}(v_{j+1}) \leq \dots \leq D_{j-1}(v_{j+s})$. Then, any reordering of ρ differs in the permutation of $v_j, v_{j+1}, \dots, v_{j+s}$ has no less total completion cost than that ordered as $v_j, v_{j+1}, \dots, v_{j+s}$.*

The above results specify that the relative order for indistinguishable simplicial nodes in any elimination graph is determined by their descendant cost and that the ones with the same descendant cost can be regarded as a "supernode". The next result identifies further the eligible set of forming a supernode and affirms their prior ordering from the nonsimplicial nodes.

THEOREM 4.5. *Assuming that in the j -th elimination step there are, after the elimination of a node v , some nonsimplicial nodes in $G^{*(j-1)}$, say u_1, u_2, \dots, u_s , become simplicial in $G^{*(j)}$. Let the maximal clique containing them be K . Then,*

(1) *the relative orders between u_1, u_2, \dots, u_s are irrelevant and,*

(2) *they should be ordered before those nonsimplicial nodes in K .*

THEOREM 4.6. *Given a nodal cost function satisfies the independent and conservative properties, our algorithm would generate an ordering of minimum completion cost among the class of perfect orderings on G^* .*

4.2. Implementation

The realization of Algorithm MinCP requires at least the following considerations: 1) The representation of the elimination graph and the testing of the

simplicial nodes; 2) The implementation of the simplicial set \mathcal{S} ; 3) The calculation of the completion cost of a simplicial node.

Elimination graph representation and simplicial node detection. As stated previously, any elimination graph can be represented as a clique structure. To simplify the discussions, we omit these material. The detailed procedure of creating the maximal cliques composed of G^* , clique representation of the elimination graph and the testing of simplicial nodes in each elimination step can be found in [9].

Simplicial set implementation. The primary requirement is an effective way to find the simplicial node with the minimum completion cost. A simple solution is using a heap structure in the sense of heapsort. The root of the heap is the node with desired property. Note that the result in Corollary 4.5 suggests the possibility of mass elimination. Thus as a node v is eliminated in step j , the set of the new simplicial nodes $\{u_1, u_2, \dots, u_s\}$ can be regarded as a supernode. Hence, only the representative node u_s is inserted into the heap structure of \mathcal{S} . This would greatly alleviate the cost of maintaining \mathcal{S} .

Calculation of the completion cost. The completion cost of a simplicial node is composed of two parts: the maximum descendant cost and the nodal cost. The nodal cost is dependant on the specification of the cost function, which is not hard to calculate. The descendant cost, as stated in Theorem 4.6, can be determined as the node becomes simplicial. Let the new simplicial nodes as a node v is eliminated in step j be u_1, u_2, \dots, u_s , which are contained within a maximal clique $K^{(j)}$. Note the relative order between u_1, u_2, \dots, u_s is irrelevant. Hence, it is appropriate to consider the nature order of u_1, u_2, \dots, u_s . Let w denote in $K^{(j)}$ the simplicial node that has the maximum completion cost among all nodes in $K^{(j)}$ that became simplicial earlier than u_1, u_2, \dots, u_s . Then it is immediately that $D_j(u_1) = D_j(u_2) = D_j(u_s) = \max\{CP(v), CP(w)\}$. Instead of keeping track of the maximum descendant cost of each simplicial node, we simply maintain this cost as the provisional cost $CP(K)$ of the maximal clique to which these simplicial nodes belong. And, in addition to inserting the simplicial node u_s into \mathcal{S} , we update the $CP(K)$. With this concept we only have to compare $CP(v)$ and $CP(K)$ to obtain the maximum descendant cost without knowing w .

THEOREM 4.7. *The complexity of Algorithm MinCP is $O(q \log q + \kappa)$, where q denote the number of maximal cliques, and κ the sum of the sizes of all of the maximal cliques in G^* .*

5. Experimental results

In this section, we present the experimental results on some test matrices from the Harwell-Boeing collection. We compare our work with Lewis, Peyton and Pothen’s [9] implementation of Jess and Kees ordering algorithm. Table 3 lists the test matrices.

Table 3. Test matrices from the Harwell-Boeing sparse matrix collection.

Key	Order	Nonzeros
BCSPWR09	1723	2394
BCSPWR10	5300	8271
BCSSTK08	1074	5943
BCSSTK13	2003	40940
BCSSTM13	2003	9970
BLCKHOLE	2132	6370
CAN 1072	1072	5686
DWT 2680	2680	11173
LSHP3466	3466	10215
PLAT1919	1919	15240

We performed the experiment on a single processor of CONVEX 3840. The programs include Liu’s multiple minimum degree ordering, GENMMD [12]; symbolic factorization, the SMBFCT of SPARSPAK [5]; minimum height ordering, follows the implementation of Lewis, Peyton and Pothen; and our minimum completion cost ordering.

Since the minimum degree ordering is sensitive to tie-breaking and in turn will affect the cost for parallel factorization, we randomly permuted all test matrices 1000 (including the original matrices) times. All methods are fed with the same randomized matrices. To our purpose we are concerned for the variance of completion cost when using Jess-Kees as the ordering method under any criterion; this value will justify the effectiveness of our method. Every randomized matrix is tested to collect the variance. The average and the best of the variances are reported in Tables 4 and 5 respectively. For most of the test matrices, the average variance is less than 10% whereas it would reach 17% for DWT2680 in the case of $\Phi 4$. The best results, however, are larger than 10% for most matrices and could reach 40% for BCSPWR10 and BCSSTM13. It is also interesting to note that the multifrontal factorization benefits the most from our ordering algorithm (see columns $\Phi 4$ and $\Phi 9$), then the submatrix-Cholesky, column-Cholesky and last the row-Cholesky factorization.

As one might expect from the purpose of ordering

Table 4. Completion cost increase (%) of Jess-Kees to MinCP (average).

key	$\Phi 2$	$\Phi 3$	$\Phi 4$	$\Phi 5$	$\Phi 6$	$\Phi 7$	$\Phi 8$	$\Phi 9$
BCSPWR09	2	2	3	1	1	2	2	2
BCSPWR10	10	6	15	4	6	10	7	16
BCSSTK08	3	1	5	1	2	3	1	5
BCSSTK13	3	0	11	0	4	3	0	11
BCSSTM13	3	0	9	0	4	3	0	9
BLCKHOLE	0	0	1	0	0	0	0	1
CAN 1072	1	0	2	0	1	1	0	2
DWT 2680	12	10	17	5	8	12	11	17
LSHP3466	1	1	2	1	1	1	1	2
PLAT1919	12	15	10	9	6	12	15	10

Table 5. Completion cost increase (%) of Jess-Kees to MinCP (best).

key	$\Phi 2$	$\Phi 3$	$\Phi 4$	$\Phi 5$	$\Phi 6$	$\Phi 7$	$\Phi 8$	$\Phi 9$
BCSPWR09	17	14	21	13	12	17	14	20
BCSPWR10	33	21	40	15	20	34	23	39
BCSSTK08	20	10	30	9	17	20	10	29
BCSSTK13	13	4	33	6	20	13	4	33
BCSSTM13	17	4	40	5	27	17	4	40
BLCKHOLE	14	6	30	6	17	14	5	30
CAN 1072	21	12	29	11	14	22	12	29
DWT 2680	22	24	33	13	16	23	24	33
LSHP3466	18	13	25	11	14	18	13	25
PLAT1919	36	37	30	24	21	37	37	30

for sparse factorization, the overhead spent on ordering should not exceed the performance gain in the factorization. Table 6 reports the CPU times of the minimum height ordering and our minimum completion ordering. As the results show, our minimum completion cost ordering consumes time approximately twofold of minimum height ordering. The overhead incurred thus by replacing minimum height ordering with our scheme is subtle as compared with the prospective improvement on the most time-consuming step—factorization.

6. Conclusions

The height of the elimination tree (parallel elimination step) has long acted as the only criterion in deriving suitable fill-preserving sparse matrix ordering for parallel factorization. Though the deficiency in adopting height as the criterion for all circumstances was well recognized, no research has succeeded in alleviating this constraint. We are the first ones to expand the ordering criterion into a more general class that reflects the various aspects in parallel factorization. We recognized that if any cost function satisfies the proposed in-

Table 6. Time (CPU seconds) to execute the minimum height (J-K) and minimum completion cost orderings (MinCP) on a single processor of CONVEX 3840.

Key	J-K	$\Phi 2$	$\Phi 3$	$\Phi 4$	$\Phi 5$	$\Phi 6$	$\Phi 7$	$\Phi 8$	$\Phi 9$
BCSPWR09	0.04	0.08	0.07	0.07	0.07	0.07	0.07	0.07	0.07
BCSPWR10	0.14	0.25	0.25	0.23	0.23	0.25	0.25	0.26	0.25
BCSSTK08	0.03	0.06	0.06	0.05	0.06	0.07	0.06	0.07	0.07
BCSSTK13	0.05	0.13	0.13	0.10	0.11	0.13	0.13	0.13	0.13
BCSSTM13	0.02	0.05	0.05	0.04	0.05	0.05	0.05	0.05	0.05
BLCKHOLE	0.05	0.10	0.11	0.08	0.09	0.10	0.10	0.11	0.10
CAN 1072	0.02	0.05	0.05	0.04	0.04	0.05	0.05	0.05	0.05
DWT 2680	0.06	0.13	0.14	0.11	0.12	0.13	0.13	0.14	0.13
LSHP3466	0.07	0.16	0.17	0.13	0.15	0.16	0.16	0.17	0.16
PLAT1919	0.04	0.09	0.10	0.08	0.08	0.09	0.09	0.10	0.09

dependent and conservative properties, a greedy based ordering scheme would generate an optimal ordering with minimum completion cost.

Recently, more effective results for parallel factorization have been achieved through a larger task model, such as the clique or supernodal tree [7] [17]. In such case ordering is not only the permutation of the nodes in the filled graph, but also affects the construction of each supernode. Different ordering may lead to a totally different set of supernodes. As to our knowledge, recent literature [2] [6] only achieved in finding ordering to minimize the height of clique tree. Methods for finding optimal orderings in terms of other more general and realistic concerns are remained unanswered. We are currently devoted to this investigation and expect to have some results in the near future.

7. Acknowledgements

The authors would like to thank John G. Lewis for many valuable suggestions and comments, especially on improving the conservative property for nodal cost function.

References

[1] B. Aspvall and P. Heggernes, *Finding minimum height elimination tree for interval graphs in polynomial time*, Bit, 34 (1994), pp. 484-509.

[2] J.R.S. Blair and B.W. Peyton, *On finding minimum-diameter clique trees*, Tech. Report ORNL/TM-11850, Oak Ridge National Laboratory, Oak Ridge, TN, 1991.

[3] 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.

[4] I.S. Duff, R.G. Grimes, and J.G. Lewis, *Sparse matrix test problems*, ACM Trans. Math. Software, 15 (1989), pp. 1-14

[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] A. Gupta and V. Kumar, *Optimally scalable parallel sparse Cholesky factorization*, in Proceedings of the Seventh SIAM Conference on Parallel Processing for Scientific Computing, SIAM (1995), pp. 442-447.

[8] J.A.G. Jess and H.G.M. Kees, *A data structure for parallel L/U decomposition*, IEEE Trans. Comput., 31 (1982), pp. 231-239.

[9] J.G. Lewis, B.W. Peyton and A. Pothén, *A fast algorithm for reordering sparse matrices for parallel factorization*, SIAM J. Sci. Stat. Comput., 10 (1989), pp. 1146-1173.

[10] W.-Y. Lin and C.-L. Chen, *Minimum completion time criterion for parallel sparse Cholesky factorization*, in Proceedings of International Conference on Parallel Processing, St. Charles, IL, 1993, pp. III 107-114.

[11] W.-Y. Lin and C.-L. Chen, *Minimum communication cost reordering for parallel sparse Cholesky factorization*, Parallel Computing, 25 (1999), pp. 943-967.

[12] J.W.-H. Liu, *Modification of the minimum degree algorithm by multiple elimination*, ACM Trans. Math. Software, 11 (1985), pp. 141-153.

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

[14] J.W.-H. Liu, *Reordering sparse matrices for parallel elimination*, Parallel Computing, 11 (1989), pp. 73-91.

[15] J.W.-H. Liu and A. Mirzaian, *A linear reordering algorithm for parallel pivoting of Chordal graphs*, SIAM J. Disc. Math., 2 (1989), pp. 100-107.

[16] 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.

[17] 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), pp. 699-713.