ON OPTIMAL REORDERINGS OF SPARSE MATRICES FOR PARALLEL CHOLESKY FACTORIZATIONS *

WEN-YANG LIN† AND CHUEN-LIANG CHEN‡

Abstract. The height of the elimination tree has long acted as the only criterion in deriving a suitable fill-preserving sparse matrix ordering for parallel factorization. Although the deficiency in adopting height as the criterion for all circumstances was well recognized, no research has succeeded in alleviating this constraint. In this paper, we extend 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. We recognize and show that if any cost function satisfies two mandatory properties, called the independence and conservation properties, a greedy ordering scheme then generates an optimal ordering with minimum completion cost. We also present an efficient implementation of the proposed ordering algorithm. Incorporating various techniques, the complexity can be improved from $O(n \log n + e)$ to $O(q \log q + \kappa)$, where n denotes the number of nodes, e the number of edges, q the number of maximal cliques and κ the sum of all maximal clique sizes in the filled graph. Empirical results show that the proposed algorithm can significantly reduce the parallel factorization cost without sacrificing much in terms of time efficiency.

Key Words. elimination tree, fill-preserving ordering, Jess and Kees algorithm, minimum completion cost, parallel Cholesky factorization, sparse matrix

AMS. 65F05, 65F50, 65Y05, 68R10

1. Introduction.

In this paper, we consider the problem of finding fill-preserving sparse matrix orderings for parallel factorization, which arises during the exploitation of parallelism in the direct solution of large sparse symmetric positive definite systems [10]. Given a large sparse symmetric and positive definite matrix A, we want to determine an ordering that is appropriate in terms of preserving the sparsity and minimizing the cost to perform its Cholesky factorization in parallel.

Jess and Kees [12] were 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 over the class of fill-preserving orderings. Some effective implementations of the Jess and Kees algorithm have been proposed [15][23]. In [22], Joseph Liu used a tree rotation heuristic to find orderings that, though not optimal, could substantially reduce the height of the elimination tree. Aspvall and Heggernes [2] also proposed an efficient algorithm to generate orderings for a minimum height elimination tree. 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. This criterion is based on the unit cost assumption for node elimination, which though simple in nature, is improper for exhibiting various factors

* A preliminary version of this paper appeared in the Proceedings of 2000 IEEE International Parallel and Distributed Processing Symposium, pp. 799–805.

† Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung, Taiwan, Email:wylin@nuk.edu.tw

‡ Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, Email:clchen@csie.ntu.edu.tw

that affect the cost of parallel factorization, such as communication overhead and the algorithmic diversity of the Cholesky form. In [16][17], we produced some examples to illustrate how the cost discrepancy may grow indefinitely. In [16][17], we also solved the problem that specifically adopts the operation count or communication message count for factoring a column/row to generate a parallel pivoting sequence with the property that it minimizes the critical completion cost for parallel factorization.

In this study, we extend the elimination tree height criterion to a more general class that is categorized as nodal cost functions satisfying two properties called the *independence* and *conservation* properties. In terms of such a class of ordering criteria, we propose a greedy algorithm to find the optimal orderings for parallel factorization. Empirical results show that on the average, the proposed minimum completion cost ordering (MinCP) reduces the parallel completion cost by up to 25% more than the well-known minimum height ordering (Jess-Kees), and by up to 66% in the best case.

The rest of this paper is organized as follows. In Section 2, we provide background material and introduce related notation. In Section 3, we 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 describes our greedy algorithm for finding optimal fill-preserving ordering. The optimality proof and the implementation details are also discussed there. Section 5 presents the experimental results on our test set. Finally, the conclusions and future work are 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 a lower triangular matrix. The solution vector x is then 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 [7]. Finding an ordering that gives the least fill is *NP*-hard [28]. Two frequently used heuristics are *minimum degree* and *nested dissection* [7].

From the study by Dongarra et al. [4], there are six algorithmic forms of Cholesky factorization obtained by permuting the three nested loops that execute the following single statement

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

Depending on which of the three indices is placed in the outer most loop, they can be classified into three basic forms [10]: *column-Cholesky*, *row-Cholesky*, and *submatrix-Cholesky*.

Here, we consider the parallel versions of these three sparse Cholesky factorizations and a more complicated variant of sparse submatrix-Cholesky called *multifrontal* factorization [5]. These four algorithms are described in Figures 1, 2, 3 and 4, respectively. In Figure 4, we let $\mathcal{F}^{(k)}$ be the frontal matrix associated with column k and $\overline{\mathcal{F}}^{(k)}$ the remaining frontal matrix after the removal of the first column. The symbols A_{*k} and L_{*k} denote column k of A and L, respectively.

for $j = 1$ **to** n **do**
  for $k = 1$ **to** $j - 1$ with $l_{jk} \neq 0$ **do**
    for $i = j$ **to** n with $l_{ik} \neq 0$ **do**
      $a_{ij} \leftarrow a_{ij} - l_{ik} l_{jk}$;
  $l_{jj} \leftarrow \sqrt{a_{jj}}$;
  for $i = j + 1$ **to** n with $a_{ij} \neq 0$ **do**
    $l_{ij} \leftarrow a_{ij} / l_{jj}$;
endfor

FIG. 1. *Sparse column-Cholesky factorization.*

2.2. Graph notation. The nonzero structure of matrix A can be conveniently represented by an undirected graph $G = (V, E)$. The n nodes, $v_1, v_2, \ldots, v_n$, correspond to the n columns/rows of the matrix, and an edge connects 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

$$\alpha : \{1, 2, \ldots, n\} \rightarrow V$$

is defined. The bijection α is called the "ordering" of G. An ordering of G corresponds to a symmetric permutation of the rows and columns of A. Hereafter, to simplify the discussion, we assume that the sparse matrix A is irreducible and has been permuted by a fill-reducing ordering, such as a 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 next equalities follow immediately from the connection of the notation with the structure of the matrix.

$$Padj(v_j, G) \equiv \{i < j \mid a_{ij} \neq 0\},$$
$$Madj(v_j, G) \equiv \{i > j \mid a_{ij} \neq 0\}.$$

The structural effect of Gaussian elimination on the matrix can easily be modeled by a sequence of *elimination* (*reduced*) *graphs* [7]: Initially, let $G^{(0)} = G$. At the ith step, where $1 \leq i \leq n$, a node v along with its incident edges are eliminated from $G^{(i-1)}$ and edges are added such that all nodes in $adj(v, G^{(i-1)})$ are pairwise adjacent. The graph structure of the resulting Cholesky factor L of A, G^*, is called the *filled graph* of A and denoted by $G^* = \bigcup_{i=0}^{n-1} G^{(i)}$.

for $i = 1$ **to** n **do**
  for $j = 1$ **to** i **do**
    for $k = 1$ **to** $j - 1$ with $l_{ik}l_{jk} \neq 0$ **do**
      $a_{ij} \leftarrow a_{ij} - l_{ik}l_{jk}$;
    if $i = j$ **then** $l_{ii} \leftarrow \sqrt{a_{ii}}$;
    else if $a_{ij} \neq 0$ **then** $l_{ij} \leftarrow a_{ij}/l_{jj}$;
endfor

FIG. 2. *Sparse row-Cholesky factorization.*

for $k = 1$ **to** n **do**
  $l_{kk} \leftarrow \sqrt{a_{kk}}$;
  for $j = k + 1$ **to** n with $a_{jk} \neq 0$ **do**
    $l_{jk} \leftarrow a_{jk}/l_{kk}$;
  for $j = k + 1$ **to** n with $l_{jk} \neq 0$ **do**
    for $i = j$ **to** n with $l_{ik} \neq 0$ **do**
      $a_{ij} \leftarrow a_{ij} - l_{ik}l_{jk}$;
endfor

FIG. 3. *Sparse submatrix-Cholesky factorization.*

for $k = 1$ **to** n **do**
  assemble column A_{*k} into $\mathcal{F}^{(k)}$;
  for each child l of k in the assembly tree **do**
    assemble the remaining frontal matrix $\overline{\mathcal{F}}^{(l)}$ into $\mathcal{F}^{(k)}$;
  apply one step Cholesky to factor the first column $\mathcal{F}^{(k)}_{*1}$ of $\mathcal{F}^{(k)}$;
  strip off $\mathcal{F}^{(k)}_{*1}$ from $\mathcal{F}^{(k)}$ to form $\overline{\mathcal{F}}^{(k)}$;
  store $\mathcal{F}^{(k)}_{*1}$ into L_{*k};
endfor

FIG. 4. *Multifrontal method.*

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 the maximal cliques $K_1, K_2, \ldots, K_q$. Following [15], the term *residual clique* refers to the resulting structure of a maximal clique that has undergone some elimination steps. We denote the residual clique of K_j after i elimination steps as $K_j^{(i)}$. Note that the residual clique may not be maximal. Indeed, when a residual clique becomes a proper subset of another residual clique, the former residual clique is not maximal and should be absorbed to maintain the clique representation of the elimination graph. Figure 5 illustrates the elimination graphs and clique structures for the first two elimination steps applied to a filled graph. Initially, $K_1^{(0)} = \{a, d\}$, $K_2^{(0)} = \{b, e, f\}$, $K_3^{(0)} = \{c, e, f\}$, $K_4^{(0)} = \{d, e, f\}$. After eliminating node a, the residual clique of K_1 is left with only one node d, which is clearly a proper subset of and absorbed by K_4. The next elimination of node b leaves nodes e and f behind in clique K_2, which is a subset of K_3 and K_4, and shall be absorbed by either of these two cliques.

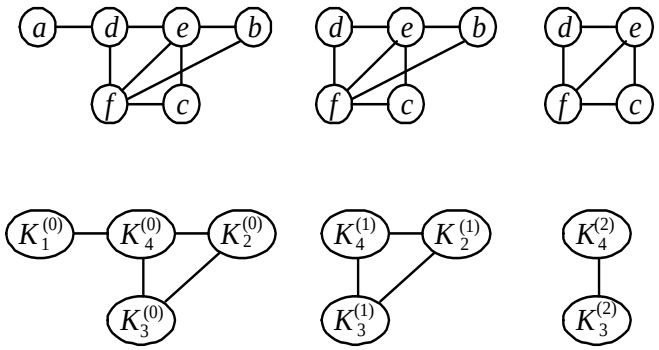

FIG. 5. *Clique representation of the elimination graph for the first two steps, where* $K_1^{(0)} = \{a, d\}$, $K_2^{(0)} = \{b, e, f\}$, $K_3^{(0)} = \{c, e, f\}$, $K_4^{(0)} = \{d, e, f\}$. *Two cliques are connected if their intersection is not empty.*

A node v in a graph G is *simplicial* if $adj(v, G)$ is a clique. A maximal clique that contains simplicial nodes is called a *simplicial 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. The next results follow immediately from these definitions.

LEMMA 2.1. *Any two simplicial nodes are either independent or indistinguishable.*

LEMMA 2.2. *A node is simplicial if and only if it belongs to only one maximal clique.*

LEMMA 2.3. *During the elimination process, a node becoming simplicial will remain simplicial until it is eliminated.*

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. In the literature [21][22], such orderings have also been called *equivalent reorderings*. In terms of graph theory, a fill-preserving ordering of A retains the structure of the filled graph G^*.

Rose [25] has shown that a filled graph is a chordal graph and always has at least one perfect ordering, i.e., an ordering with no fill. For example, the ordering used to generate the filled graph is a perfect ordering of the filled graph. It is interesting to note that if A is reordered by a perfect ordering on G^*, the resulting filled graph $\tilde{G}^*$ is a subset of G^* [21]. Some of the filled edges in G^* may not appear in $\tilde{G}^*$. That is, a perfect ordering on G^*, as an ordering of A, is at least as good as the original fill-preserving ordering of A in terms of fill and computation. Therefore, finding a fill-preserving ordering for matrix A can be regarded as finding a perfect ordering of the filled graph G^*.

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

LEMMA 2.4. *An ordering $(v_1, v_2, \ldots, v_n)$ on G^* is perfect if and only if v_j is simplicial in $G^{*(j-1)}$, $1 \leq j \leq n$.*

3. Computation model and cost specification.

3.1. The elimination tree model. The *elimination tree* model, which was introduced in [12], has been shown to be useful 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 $parent(v_i) = \min\{v_k \mid k > i \text{ and } l_{ki} \neq 0\}$. The elimination tree captures the column/row dependencies of the Cholesky factor [20]. Short elimination trees imply less interdependency and thus more parallelism. Liu [20] illustrated that the elimination tree model is appropriate for analyzing the three versions of Cholesky factorization. Duff and Reid [5] also used the elimination tree as a structure in their multifrontal factorization. We thus concentrate on this computation model.

Let $T[v]$ denote the subtree of $T(A)$ rooted at node v. Some useful properties of the elimination trees in [12] are given below.

LEMMA 3.1. *If $l_{jk} \neq 0$ and $k < j$, then the node v_k is a descendant of v_j in $T(A)$.*

This lemma implies that the root vertex is v_n.

LEMMA 3.2. *Assume $T[v]$ and $T[u]$ are two disjoint subtrees. Then in G^* there is no edge (x, y) with $x \in T[v]$ and $y \in T[u]$.*

Consider a perfect ordering α on G^*. Let $\Phi_\alpha(v)$ specify the cost of eliminating a node v in G^*_α with respect to the Cholesky factorization. Since the elimination tree exhibits the column dependencies of the Cholesky factor, it is natural to define the nonnegative cost (called *completion cost*) to complete the parallel elimination at node v as the critical weighted path on the subtree $T[v]$. More precisely, let $CP_\alpha(\ )$ denote

the completion cost function. We have the following recursive definition

$$CP_\alpha(v) = \begin{cases} \Phi_\alpha(v), & \text{if } v \text{ is a leaf node,} \\ \Phi_\alpha(v) + \max\{CP_\alpha(u) \mid u \text{ is a child of } v\}, & \text{otherwise.} \end{cases}$$

The total completion cost, CP_α, to eliminate the entire graph or factorize the corresponding matrix is then equal to $CP_\alpha(v_n)$. We omit the "ordering" subscript when it is clear from the context.

3.2. Independence and conservation properties. Given a filled graph, the work of finding an optimal perfect ordering to minimize the parallel factorization time is not at all trivial; current approaches do not take into consideration various algorithmic forms of factorization. Our previous study [17] provides us with a foundation to work from. We observed that a greedy approach seems promising to yield an optimal perfect ordering under various considerations. We further found that the conditions assuring the optimality of a greedy ordering algorithm can be clarified as two important properties on the cost specification.

Let ρ be a perfect ordering on G^*:

$$\rho : v_1, v_2, \ldots, v_n,$$

such that nodes v_j and v_{j+1} are both simplicial in $G_\rho^{*(j-1)}$. Then the ordering

$$\tilde{\rho} : v_1, v_2, \ldots, v_{j-1}, v_{j+1}, v_j, v_{j+2}, \ldots v_n$$

that differs from ρ only in positions j and $j+1$ is also a perfect ordering. We define a class Ω of cost functions that satisfies the following independence and conservation properties. That is, $\Omega = \{\Phi : \Phi$ satisfies the independence and conservation 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 independence 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 conservation property if, when v_j and v_{j+1} are interchanged in the elimination order, the sum of the cost of v_j and v_{j+1} is unchanged. That is,*

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

It is worthwhile to point out that when the two simplicial nodes v_j and v_{j+1} belong to different maximal cliques, the two subtrees $T[v_j]$ and $T[v_{j+1}]$ are disjoint. The cost independence property thus constrains the class Ω to preserve the subtree independence property. The conservation property further restricts our attention to the functions that reflect the graph indistinguishability of any nodes within the same simplicial clique, a principle useful in many aspects of the sparse matrix computations [10].

3.3. Example cost functions in Ω. There are various factors affecting the specification of the cost function for eliminating nodes in parallel. Some typical factors are parallel architecture: distributed- or shared-memory, the number of processors available, the task granularity and scheduling strategy, and the algorithmic form of the Cholesky factorization. On the basis of the elimination tree model, we considered the parallel sparse column-, row-, submatrix-Cholesky and multifrontal methods 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, and 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 interconnection and routing topology.

It is difficult to define one cost function that is generally good for most cases. Instead, we chose to specify a cost function for each case. As an example, we considered 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 function is described in Table 2, where the notation $K_{v_j}^{(i)}$ denotes the residual set of the maximal clique K_{v_j} in which a node v_j becomes simplicial. The derivation of these specifications is omitted here. We only show in Figure 6 an example for $\Phi 2$. The details can be found in [18].

TABLE 1
A category of node cost functions.

		Column	Row	Submatrix	Multifrontal
computation	unit	$\Phi 1$	$\Phi 1$	$\Phi 1$	$\Phi 1$
	multiplications	$\Phi 2$	$\Phi 3$	$\Phi 4$	$\Phi 4$
communication	number of messages	$\Phi 5$	$\Phi 5$	$\Phi 5$	$\Phi 6$
	volume	$\Phi 7$	$\Phi 8$	$\Phi 7$	$\Phi 9$

Note that function $\Phi 1$ corresponds to the case for unit cost. That is, the criterion for using elimination tree height is a special case of the proposed class. It is not hard to verify that each function satisfies the independence and conservation properties. When v_j and v_{j+1} are independent in $G_\rho^{*(j-1)}$, the assertion is evident because the interchange of v_j and v_{j+1} doe not change their prior (monotone) adjacent sets as well as the structure of the elimination tree. On the other hand, when v_j and v_{j+1} are indistinguishable in $G_\rho^{*(j-1)}$, the interchange of v_j and v_{j+1} makes v_{j+1} become a member of v_j's prior adjacent set and v_j a member of v_{j+1}'s monotone adjacent set. Although this would change the nodal costs of v_j and v_{j+1}, the cost variations are the same; thus the sum of the cost remains unchanged.

As an illustration, consider the filled graph in Figure 5 again. Figures 7 and 8 illustrate the independence and conservation properties for the nodal cost function $\Phi 2$,

TABLE 2
Specification of the cost functions in Table 1.

$\Phi 1$	1
$\Phi 2$	$\sum_{v_j \in Padj(v_i) \cup \{v_i\}} \lvert K_{v_j}^{(i-1)} \rvert$
$\Phi 3$	$\sum_{v_j \in Padj(v_i) \cup \{v_i\}} (\lvert K_{v_j}^{(j-1)} \rvert - \lvert K_{v_j}^{(i)} \rvert)$
$\Phi 4$	$\frac{1}{2} \lvert K_{v_i}^{(i-1)} \rvert (\lvert K_{v_i}^{(i-1)} \rvert + 1)$
$\Phi 5$	$Pdeg(v_i)$
$\Phi 6$	$\sum_{v_j \in child(v_i)} Mdeg(v_j)$
$\Phi 7$	$\sum_{v_j \in Padj(v_i)} \lvert K_{v_j}^{(i-1)} \rvert$
$\Phi 8$	$\sum_{v_j \in Padj(v_i)} (\lvert K_{v_j}^{(j-1)} \rvert - \lvert K_{v_j}^{(i-1)} \rvert)$
$\Phi 9$	$\sum_{v_j \in child(v_i)} \frac{1}{2} \lvert K_{v_j}^{(i-1)} \rvert (\lvert K_{v_j}^{(i-1)} \rvert + 1)$

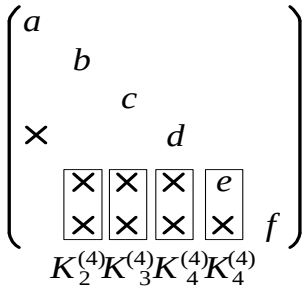

FIG. 6. *Illustration of nodal cost* $\Phi 2$.

which measures the multiplicative operations for column-Cholesky factorization. In Figure 7, c and d are two independent simplicial nodes after the first two elimination steps. The nonzeros surrounded by squares or circles represent the contribution to $\Phi 2(c)$ or $\Phi 2(d)$ respectively. As the picture shows, the effect of interchanging c and d is nothing more than a relabelling. In Figure 8, d and e are indistinguishable simplicial nodes after elimination step 3. It can be observed that $\Phi 2(d) + \Phi 2(e)$ equals $4 + 8 = 12$ before the interchange and remains $5 + 7 = 12$ after the interchange.

OBSERVATION. *A linear combination of any two cost functions in Table 1 with the form* $\Phi i + \lambda \Phi j$ *is also in* Ω*, where* λ *denotes a constant.*

For example, assume that a communication-to-computation cost ratio is λ. The cost function for eliminating a node under column-Cholesky factorization can be denoted as $\Phi 2 + \lambda \Phi 5$ or $\Phi 2 + \lambda \Phi 7$, a more realistic form.

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 is G^*. Our intention is to find a perfect ordering $\sigma : \{1, 2, \ldots, n\} \rightarrow V$ on G^* to minimize the completion cost.

Lemma 2.4 gives us the basic guideline—use the elimination process model. In each

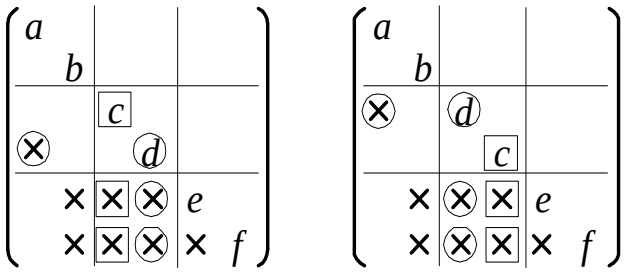

FIG. 7. *Illustration of the cost independent property.*

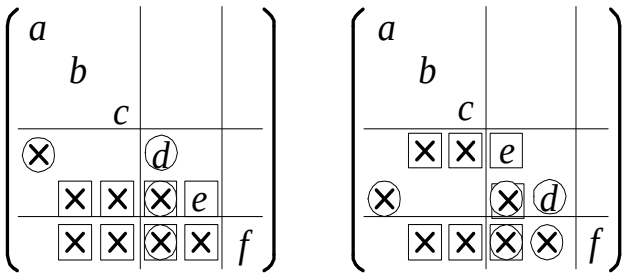

FIG. 8. *Illustration of the cost conservative property.*

elimination step, the nodes that are simplicial are identified. Then based on a greedy paradigm, one of the simplicial nodes with the minimum completion cost as defined in Section 3 is chosen to be labeled and eliminated next. This process continues until all nodes are eliminated. This greedy ordering strategy is summarized in Algorithm 1.

ALGORITHM 1. *A greedy fill-preserving ordering algorithm.*
$G^{(0)} \leftarrow G^*$;
for $i = 1$ **to** n **do**
  $\mathcal{S} \leftarrow$ the set of all simplicial nodes in $G^{(i)}$;
  pick a simplicial node v for which $CP(v)$ is minimized over $\mathcal{S}$;
  $\sigma(i) \leftarrow v$;
  $G^{(i)} \leftarrow G^{(i-1)} \backslash \{v\}$;
endfor

Consider the step for picking the next eliminated node. There are some points deserved further clarification. First, when node v is chosen to be eliminated in the ith elimination step, we have to calculate its completion cost with the knowledge of an incomplete ordering $\sigma(1), \sigma(2), ..., \sigma(i)$. This seems to be inconsistent with the defin-

ition given in Section 3.1, where the completion cost function is determined from the elimination tree corresponding to a complete ordering. But recall that the completion cost of node v equals to the critical weighted path on the subtree rooted at v, and all descendants of v are ordered earlier than v. Thus, knowing the incomplete ordering $\sigma(1), \sigma(2), ..., \sigma(i)$ is sufficient to determine the structure of subtree $T[v]$.

Second, to choose the next node for elimination, we have to evaluate the completion cost of each simplicial node, which however cannot be worked out until the node is actually eliminated. To resolve the uncertainty about node ordering, we tentatively regard each simplicial node as the next eliminated node and figure out its completion cost. The computed cost thus is a provisional completion cost. The node with the minimum provisional completion cost is eliminated next and its provisional completion cost becomes its actual completion cost.

For illustration, consider the graph in Figure 5. After eliminating nodes a, b, and c, the remaining three nodes, d, e, and f, in the graph are simplicial. Assume the nodal cost is $\Phi 2$. If d is regarded as the next eliminated node, the completion cost of d is $CP(a) + \Phi 2(d) = 2 + 3 = 5$. Similar calculations yield that both e and f have the same provisional completion cost, $3 + 3 = 6$. Hence, node d is eliminated next.

4.2. Minimum completion cost property. In this subsection, we will show that given a filled graph G^*, the ordering obtained with Algorithm 1 minimizes the completion cost over all perfect orderings of G^*.

To facilitate the discussion, we consider two cases: the relative order between independent simplicial nodes and that between indistinguishable simplicial nodes. More specifically, let $\rho : v_1, v_2, \ldots, v_n$ be any perfect ordering in which nodes v_j and v_{j+1} are both simplicial in $G^{*(j-1)}$. Then a reordering of ρ, which differs only in the j-th and $j+1$-st positions is also a perfect ordering, i.e., $\widetilde{\rho} : v_1, v_2, \ldots, v_{j-1}, v_{j+1}, v_j, v_{j+2}, \ldots, v_n$. We will show the effect the interchange of v_j and v_{j+1} has on the completion cost, for each of the two different cases that v_j and v_{j-1} are independent or indistinguishable, and extend the result to a more general case for interchanging a set of consecutive simplicial nodes $v_j, v_{j+1}, \ldots, v_{j+s}$ in $G^{*(j-1)}$.

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

Proof. Since v_j and v_{j+1} are independent, $T[v_j]$ and $T[v_{j+1}]$ are disjoint, and the elimination tree remains the same after the interchanging of v_j and v_{j+1}. The statement then follows from the assumption that Φ satisfies the independence property. ∎

COROLLARY 4.2. *Assume that $v_j, v_{j+1}, \ldots, v_{j+s}$ is an independent set of simplicial nodes in $G^{*(j-1)}$. Then, the relative order between $v_j, v_{j+1}, \ldots, v_{j+s}$ 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

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

and $D_j(v) = 0$ when $adj(v, G^*) \cap \{v_1, v_2, \ldots, v_j\} = \emptyset$. We do not 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 simplicial and belong to the same maximal clique in $G^{*(j-1)}$ and $D_{j-1}(v_j) \geq D_{j-1}(v_{j+1})$, then $CP_{\tilde{\rho}} \leq CP_\rho$.*

Proof. Clearly, an edge exists between v_j and v_{j+1}. By the elimination tree definition, v_{j+1} must be the parent of v_j in the case of ordering ρ. Therefore,

$$\begin{aligned} CP_\rho(v_{j+1}) &= \Phi_\rho(v_{j+1}) + \max\{CP_\rho(v_j), D_{j-1}(v_{j+1})\}, \\ &= \Phi_\rho(v_{j+1}) + \max\{\Phi_\rho(v_j) + D_{j-1}(v_j), D_{j-1}(v_{j+1})\}, \\ &= \Phi_\rho(v_{j+1}) + \Phi_\rho(v_j) + D_{j-1}(v_j). \end{aligned}$$

In ordering $\tilde{\rho}$, v_j becomes the parent of v_{j+1}. We have

$$\begin{aligned} CP_{\tilde{\rho}}(v_j) &= \Phi_{\tilde{\rho}}(v_j) + \max\left\{CP_{\tilde{\rho}}(v_{j+1}), D_{j-1}(v_j)\right\}, \\ &= \Phi_{\tilde{\rho}}(v_j) + \max\left\{\Phi_{\tilde{\rho}}(v_{j+1}) + D_{j-1}(v_{j+1}), D_{j-1}(v_j)\right\}, \end{aligned}$$

It is immediately clear that $CP_{\tilde{\rho}}(v_j)$ will be $\Phi_{\tilde{\rho}}(v_j) + D_{j-1}(v_j)$ or $\Phi_{\tilde{\rho}}(v_j) + \Phi_{\tilde{\rho}}(v_{j+1}) + D_{j-1}(v_{j+1})$. Both are no greater than $CP_\rho(v_{j+1})$ due to the conservation property and the condition that $D_{j-1}(v_j) \geq D_{j-1}(v_{j+1})$. The lemma then follows. □

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

COROLLARY 4.5. *Assume that $v_j, v_{j+1}, \ldots, v_{j+s}$ is an indistinguishable set of simplicial nodes in $G^{*(j-1)}$ and $D_{j-1}(v_j) = D_{j-1}(v_{j+1}) = \cdots = D_{j-1}(v_{j+s})$. Then, any reordering of ρ that differs only in the permutation of $v_j, v_{j+1}, \ldots, v_{j+s}$ has the same total completion cost.*

THEOREM 4.6. *If a nodal cost function satisfies the independence and conservation properties, Algorithm 1 generates a minimum completion cost ordering among the class of perfect orderings on G^*.*

Proof. Let $\sigma : u_1, u_2, \ldots, u_n$ be the ordering obtained by Algorithm 1 and $\rho : v_1, v_2, \ldots, v_n$ be an optimal ordering. Clearly, both σ and ρ are perfect orderings. We will show that ρ can be transformed into σ without increasing the completion cost. Hence, σ also is optimal.

Assume that $\sigma \neq \rho$ and k is the least index such that $u_k \neq v_k$. Since both σ and ρ are perfect orderings, u_k and v_k must be simplicial in $G_\sigma^{*(k-1)}$ as well as $G_\rho^{*(k-1)}$ by Lemma 2.4. Furthermore, let $v_j = u_k$ for some j, $k < j \leq n$, and $\tilde{\rho}$ be a reordering of ρ such that v_j is ordered immediately before v_k whereas other nodes remain unchanged, i.e., $\tilde{\rho} : v_1 = u_1, \ldots, v_{k-1} = u_{k-1}, v_j, v_k, \ldots, v_{j-1}, v_{j+1}, \ldots, v_n$. We want to show that $\tilde{\rho}$ can be obtained from a sequence of reorderings of ρ such that v_j is shifted left one position each time. The sequence is defined as follows:

$$\begin{aligned} \omega^{(0)}(\rho) &= \rho, \\ \omega^{(i)}(\rho) &= \omega(\omega^{(i-1)}(\rho)) \\ &= \omega(v_1, \ldots, v_{j-i-1}, \boldsymbol{v_{j-i}}, \boldsymbol{v_j}, v_{j-i+1}, \ldots, v_{j-1}, v_{j+1}, \ldots, v_n) \\ &= v_1, \ldots, v_{j-i-1}, \boldsymbol{v_j}, \boldsymbol{v_{j-i}}, v_{j-i+1}, \ldots, v_{j-1}, v_{j+1}, \ldots, v_n, \qquad 1 \leq i \leq j-k. \end{aligned}$$

Note that v_j is simplicial in $G^{*(k-1)}$. It follows from Lemma 2.3 that v_j remains simplicial in $G^{*(l)}$, for $l \geq k$. Hence the sequence of reorderings of ρ defined above produce a perfect ordering.

Consider $\omega^{(i)}(\rho)$. For simplicity, let $\gamma = \omega^{(i-1)}(\rho)$ and $\widetilde{\gamma} = \omega^{(i)}(\rho)$. To show that $CP_{\widetilde{\gamma}} \leq CP_{\gamma}$, we only have to deliberate, according to Lemma 4.3, on the case that v_j and v_{j-i} belong to the same maximal clique in $G_{\gamma}^{*(j-i-1)}$ and show that $D_{j-i-1}(v_j) \leq D_{j-i-1}(v_{j-i})$.

There are two cases to consider. Suppose first that v_{j-i} is simplicial in $G_{\sigma}^{*(k-1)}$. Since the algorithm chooses v_j at step k, it follows that $D_{k-1}(v_j) \leq D_{k-1}(v_{j-i})$. Since both v_j and v_{j-i} are simplicial in the same maximal clique of $G_{\sigma}^{*(k-1)}$, it follows that $D_{j-i-1}(v_{j-i}) \geq D_{j-i-1}(v_j)$, which is the desired result. On the other hand, suppose that v_{j-i} is not simplicial in $G_{\sigma}^{*(k-1)}$. Then v_{j-i} has at least one descendant in $G_{\sigma}^{*(k-1)}$ and at least one of these descendants, say w, has to be simplicial in $G_{\sigma}^{*(k-1)}$. Again, from the choice of the algorithm at step k, we know that $D_{k-1}(v_j) \leq D_{k-1}(w)$. The ancestor of w right below v_{j-i} is adjacent to v_{j-i}. Since cost increases as one goes up the elimination tree, $D_{k-1}(v_j) \leq D_{j-i-1}(v_{j-i})$. Clearly, if $D_{j-i-1}(v_j) = D_{k-1}(v_j)$, then $D_{j-i-1}(v_{j-i}) \geq D_{j-i-1}(v_j)$. If $D_{j-i-1}(v_j) > D_{k-1}(v_j)$, then some simplicial node u in the same maximal clique containing v_j and v_{j-i} is ordered earlier than $j-i$ and after $k-1$, and $CP(u) = D_{j-i-1}(v_j)$. Since v_{j-i} is also adjacent to u, it follows that $D_{j-i-1}(v_{j-i}) \geq D_{j-i-1}(v_j)$, which completes the proof of the result. □

4.3. Enhancements. Though Algorithm 1 generates a minimum completion cost ordering, it takes more time than necessary to obtain the result. The bottleneck lies in the step that determines the simplicial node with minimum completion cost. From the time a node becomes simplicial, we have to reevaluate its provisional completion cost in all successive steps that eliminate a neighboring simplicial node until the node is eliminated. We will show in this subsection that the completion cost reevaluation is not essential. Here we will also demonstrate how we can derive a supernode-based ordering strategy.

LEMMA 4.7. *The ordering obtained from Algorithm 1 is in nondecreasing order of the completion cost.*

Proof. The lemma follows from the fact that a node v deleted from $\mathcal{S}$ has the minimum completion cost among all nodes in $\mathcal{S}$ and the new simplicial nodes to be inserted into $\mathcal{S}$ are ancestors of v. □

THEOREM 4.8. *Assume that in the j-th elimination step of applying Algorithm 1 on G^* there are, after the elimination of a node v, some nonsimplicial nodes in $G^{*(j-1)}$, say u_1, u_2, ..., u_s, that become simplicial in a maximal clique K in $G^{*(j)}$. Then*

(1) $u_1, u_2, \ldots, u_s$ can be ordered before the nonsimplicial nodes in K, and

(2) the relative order between $u_1, u_2, \ldots, u_s$ is irrelevant.

Proof. To prove the first statement, we assume that a nonsimplicial node w in $K - \{u_1, u_2, \ldots, u_s\}$ is ordered before or between u_1, u_2, ..., u_s. More specifically, let w be eliminated in step k and ordered between u_{i-1} and u_i, for $1 \leq i \leq s$ and $u_0 = v$. The fact that w is nonsimplicial implies that w will have some descendant ordered later than the children of u_i excluding w. It follows that $D_{k-1}(w) \geq D_{k-1}(u_i)$. Hence, w can

be ordered after u_i according to Lemma 4.3. This interchange can be repeated until w is ordered after $u_1, u_2, \ldots, u_s$. Hence the first statement follows.

Next, note that the completion costs of all nodes adjacent to $u_1, u_2, \ldots, u_s$ and eliminated before v are, according to Lemma 4.7, no greater than that of v. It follows that $u_1, u_2, \ldots, u_s$ share the same descendant cost immediately after the elimination of v. The only neighbors of $u_1, u_2, \ldots, u_s$ eliminated subsequently are simplicial nodes in K. It follows that $u_1, u_2, \ldots, u_s$ continue to share the same descendant cost until the first u_i is eliminated. At that point, it follows from Corollary 4.5 that the relative order between $u_1, u_2, \ldots, u_s$ is irrelevant. □

As a result, during the progress of greedy elimination, the relative order for indistinguishable simplicial nodes is determined once they became simplicial. The elimination step at which nodes become simplicial thus divides a maximal clique into different classes: several sets of simplicial nodes and one set of nonsimplicial nodes, and the simplicial nodes classified into the same set can be regarded as a "supernode." This relative ordering posed on the simplicial nodes and nonsimplicial nodes also resolves the uncertainty about the children set of a simplicial node, knowledge of which is required for evaluating its completion cost. Algorithm 2 describes a refinement of Algorithm 1 incorporating this supernode-based elimination without reevaluating the completion cost.

ALGORITHM 2. *A supernodal variant of Algorithm 1.*

$G^{(0)} \leftarrow G^*$;
let $K_1, K_2, \ldots, K_q$ be the maximal cliques of G^*;
for each $K_j, 1 \le j \le q$ **do**
  $U \leftarrow$ the set of simplicial nodes in K_j;
  denote U as a supernode and compute its completion cost;
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{U\}$;
endfor
$i \leftarrow 1$;
while $\mathcal{S} \neq \emptyset$ **do**
  pick a supernode U from $\mathcal{S}$ with $\min_{U \in \mathcal{S}} CP(U)$;
  for each node $v \in U$ **do**
    $\sigma(i) \leftarrow v$;
    $G^{(i)} \leftarrow G^{(i-1)} \backslash \{v\}$;
    $i \leftarrow i + 1$;
  endfor
  $\mathcal{S} \leftarrow \mathcal{S} \backslash \{U\}$;
  $U \leftarrow$ the set of new simplicial nodes in $G^{(i)}$;
  denote U as a supernode and compute its completion cost;
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{U\}$;
endwhile

4.4. Implementation. The realization of Algorithm 2 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 $\mathcal{S}$; and 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 this material. The detailed procedure for creating the maximal cliques composing G^*, the clique representation of the elimination graph and the testing of simplicial nodes in each elimination step can be found in [15].

Simplicial set implementation. The primary requirement is an effective way to find the simplicial supernode with the minimum completion cost. A simple solution is using a heap structure in the sense of a heapsort. The root of the heap is the node with the desired property. Note that the result in Theorem 4.8 suggests the possibility of mass elimination. Thus, as a node v is eliminated in step j, the set of the new simplicial nodes $\{u_1, u_2, \ldots, u_s\}$ can be regarded as a supernode. Hence, only the representative node, say u_s, is inserted into the heap structure of $\mathcal{S}$. This greatly alleviates the cost of maintaining $\mathcal{S}$.

Completion cost calculation. The completion cost of a simplicial node is composed of two parts: the maximum descendant cost and the nodal cost. The nodal cost depends on the specification of the cost function, which is not hard to calculate, but, if not simplified, will dominate the computation of the whole algorithm. In particular, consider the nodal cost functions $\Phi 2$, $\Phi 3$, $\Phi 5$, $\Phi 7$ and $\Phi 8$. Calculating these functions involves an accumulation over the prior (or monotone) adjacent set, yielding a complexity of $O(e)$ for all simplicial nodes throughout the algorithm, where e is the number of edges in G^*. The computation can be reduced if the cost functions are specified in terms of the simplicial clique notion. For this purpose, we introduce the notation $\eta_K^{(j)}$, which denotes the set of nodes ordered no later than v_j and whose simplicial clique is K. That is,

$$\eta_K^{(j)} = \{v_i | K_{v_i} = K, \text{ for } i \le j\}.$$

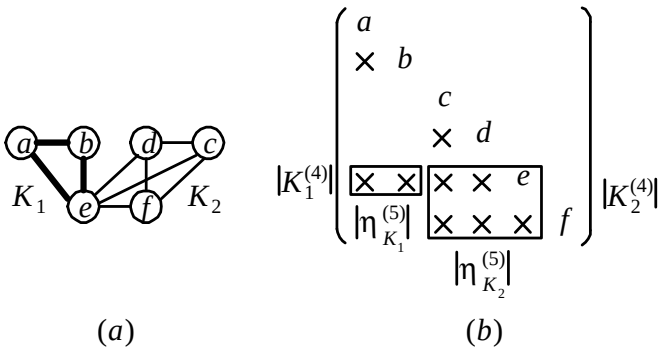

FIG. 9. *An example filled graph and the corresponding filled matrix.*

As an illustration, consider the graph in Figure 9(a). There are two maximal cliques $K_1 = \{a, b, e\}$ and $K_2 = \{c, d, e, f\}$. Assuming that the nodes are ordered as a, b, c, d, e, f, we can derive

$$\begin{array}{ll} \eta_{K_1}^{(1)} = \{a\} & \eta_{K_2}^{(1)} = \emptyset \\ \eta_{K_1}^{(2)} = \{a, b\} & \eta_{K_2}^{(2)} = \emptyset \\ \eta_{K_1}^{(3)} = \{a, b\} & \eta_{K_2}^{(3)} = \{c\} \\ \eta_{K_1}^{(4)} = \{a, b\} & \eta_{K_2}^{(4)} = \{c, d\} \\ \eta_{K_1}^{(5)} = \{a, b\} & \eta_{K_2}^{(5)} = \{c, d, e\} \\ \eta_{K_1}^{(6)} = \{a, b\} & \eta_{K_2}^{(6)} = \{c, d, e, f\}. \end{array}$$

Let $\mathcal{M}(v_i)$ denote the set of maximal cliques containing v_i in G^*. Note that $\mathcal{M}(v_i)$ also represents the set of simplical cliques of the nodes in $Padj(v_i) \cup \{v_i\}$, i.e.,

$$\mathcal{M}(v_i) = \bigcup_{v_j \in Padj(v_i) \cup \{v_i\}} K_{v_j}.$$

In short, the nodes having the same simplicial cliques forms a supernode. Using this representation, the nodal cost $\Phi 2$ can be rewritten as

$$\sum_{K \in \mathcal{M}(v_i)} |\eta_K^{(i)}||K^{(i-1)}|.$$

For example, consider the nodal cost $\Phi 2$ of node e in Figure 9(a). The set of maximal cliques containing e is $\{K_1, K_2\}$, $\eta_{K_1}^{(5)} = \{a, b\}$, $\eta_{K_2}^{(5)} = \{c, d, e\}$, $K_1^{(4)} = \{e\}$, and $K_2^{(4)} = \{e, f\}$. Hence, $\Phi 2(e) = |\eta_{K_1}^{(5)}||K_1^{(4)}| + |\eta_{K_2}^{(5)}||K_2^{(4)}| = 8$, which is depicted in Figure 9(b). Similar transformation can be applied to $\Phi 3$, $\Phi 5$, $\Phi 7$, and $\Phi 8$. As for $\Phi 6$ and $\Phi 9$, the set $child(v_i)$ can be transformed to the set $\mathcal{CM}(v_i)$ so as to facilitate the supernode representation, where $\mathcal{CM}(v_i)$ represents the set of simplicial cliques of the nodes in $child(v_i)$. Table 3 shows the modified specifications for the cost functions in Table 2. Using this modification, the computation for nodal cost calculation will be reduced to $O(\kappa)$, where κ represents the sum of the sizes of all of the maximal cliques in G^*.

TABLE 3
Modified specifications for cost functions in Table 2.

$\Phi 2$	$\sum_{K \in \mathcal{M}(v_i)} \|\eta_K^{(i)}\|\|K^{(i-1)}\|$
$\Phi 3$	$\sum_{K \in \mathcal{M}(v_i)} \frac{1}{2}\|\eta_K^{(i)}\| \left(2(\|K^{(0)}\| - \|K^{(i)}\|) - \|\eta_K^{(i)}\| + 1\right)$
$\Phi 5$	$\sum_{K \in \mathcal{M}(v_i)} \|\eta_K^{(i-1)}\|$
$\Phi 6$	$\sum_{K \in \mathcal{CM}(v_i)} \|\eta_K^{(i-1)}\|$
$\Phi 7$	$\sum_{K \in \mathcal{M}(v_i)} \|\eta_K^{(i-1)}\|\|K^{(i-1)}\|$
$\Phi 8$	$\sum_{K \in \mathcal{M}(v_i)} \frac{1}{2}\|\eta_K^{(i-1)}\| \left(2(\|K^{(0)}\| - \|K^{(i-1)}\|) - \|\eta_K^{(i-1)}\| + 1\right)$
$\Phi 9$	$\sum_{K \in \mathcal{CM}(v_i)} \frac{1}{2}\|K^{(i-1)}\|(\|K^{(i-1)}\| + 1)$

The completion cost, as stated in Theorem 4.8, can be determined as the node becomes simplicial. Let the new simplicial nodes, as a node v is eliminated in step

j, be $u_1, u_2, \ldots, u_s$ and all of them are contained within a maximal clique $K^{(j)}$. Note that the relative order between $u_1, u_2, \ldots, u_s$ is irrelevant. Hence, it is appropriate to consider the natural order of $u_1, u_2, \ldots, u_s$. 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 $u_1, u_2, \ldots, u_s$. Then it is immediately clear that $D_j(u_1) = D_j(u_2) = \ldots = D_j(u_s) = \max\{CP(v), CP(w)\}$. 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. In addition to inserting the simplicial node u_s into $\mathcal{S}$, we update $CP(K)$. With this concept we only have to compare $CP(v)$ and $CP(K)$ to obtain the maximum descendant cost without knowing w.

THEOREM 4.9. *The complexity of Algorithm 2 is $O(q \log q + \kappa)$, where q denotes the number of maximal cliques and κ the sum of the sizes of all of the maximal cliques in G^*.*

Proof. It has been shown in [15] that the creation of maximal cliques and the representation of G^* can be completed in $O(n + \kappa)$ time, where $\kappa = \sum_{i=1}^{q} |K_i|$ and n denotes the number of nodes in G^*. Since each supernode becomes new simplicial only upon the absorption of a maximal clique and each insertion to the simplicial set $\mathcal{S}$ takes $O(\log q)$ time, the maintenance of the simplicial set $\mathcal{S}$ through the whole process consumes at most $O(q \log q)$ time. The cost for computing the nodal cost is constant or proportional to $Pdeg(v)$ or $|child(v)|$, depending on the specification of the cost function. For n nodes, the total cost would be $O(e)$ in the worst case, where e denote the number of edges in G^*. Note that in our implementation the calculation is indeed performed via the supernode concept. The cost is thus reduced to $O(\kappa)$. Hence, the overall complexity is $O(q \log q + \kappa)$. □

5. Experimental results. In this section, we present the experimental results on some test matrices from the collection at the University of Florida[1]. Table 4 lists the name, a description, the order, and the number of nonzeros in the lower triangular part of the test matrices.

All experiments were performed on a SGI Origin 3800 system, with 32 processors. We first used Liu's multiple minimum degree algorithm MMD [19] to reorder each of the matrices and performed symbolic factorization to obtain the corresponding filled matrix. Thereafter we applied the Jess and Kees algorithm (following the leading implementation in [15]) and our minimum completion cost ordering algorithm, respectively, to each of the reordered matirces. 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 100 times (including the original one). All methods were run on the same randomized matrices. For our purpose, we are concerned with the completion cost difference when using Jess-Kees as the ordering method under any criterion. This value will justify the effectiveness of our method. Every randomized matrix was tested to collect the difference.

[1] Available from `http://www-pub.cise.ufl.edu/~davis/sparse`

The mean and the best of the differences are reported in Tables 5 and 6, respectively. For most of the test matrices, the mean difference is less than 10% whereas it reaches 18% for BCSSTK30, BCSSTK35, and CFD1 in the case of $\Phi 4$ and $\Phi 9$. The result for best difference, however, is larger than 15% for most matrices, and reaches 41% and 66% for BCSSTK35 and SKIRT, respectively. Note that the results in the first three columns are quite similar to those in the last three columns. This is because the formulations of $\Phi 2$, $\Phi 3$, and $\Phi 4$ are similar to $\Phi 7$, $\Phi 8$, and $\Phi 9$, respectively. 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.

It is well known that MMD will lead to elimination trees that are not well balanced. So, most good orderings used today are hybrids of nested dissection and minimum degree [1] [11] [14] [24]. We thus conducted another experiment, using one of the leading hybrid ND/MD orderings, METIS [13], in place of MMD, to evaluate the gains of our ordering. The results are shown in Tables 7 and 8.

It is not surprising that the gains decrease for most of the test matrices, which corroborates the result in the literature that ND produces more balanced elimination trees than MMD and leads to more parallelism. For matrices derived from complex geometries, however, it is known that ND does not necessarily produce good separators and/or result in well balanced elimination trees. In this case, one can expect large reductions in completion cost through our algorithm. Matrices SHUTTLE, STRUCT3, and FINAN512 highlight this phenomenon; all exhibit more cost reductions when they are initially ordered by METIS than by MMD, measured by either the mean or the best difference. Other matrices in our test set that partially exhibit this phenomenon include BARTH4, BARTH5, BCSSTK30, BCSSTK37, and PWT. For these matrices, we observe that, in the case of multifrontal factorization, the performance gains obtained with initial orderings from METIS are larger than those from MMD. Similar to the case when the initial orderings are produced by MMD, the multifrontal factorization benefits the most from our ordering algorithm, then the submatrix-Cholesky, column-Cholesky, and last the row-Cholesky factorization.

As one might expect, the overhead spent on ordering should not exceed the performance gain in the factorization. Table 9 reports the CPU times for the minimum height ordering and our minimum completion ordering. As the results show, our minimum completion cost ordering takes approximately 2–4 times more time than minimum height ordering. The overhead incurred by replacing minimum height ordering with our scheme is small as compared with the prospective improvement in the most time-consuming step—factorization.

6. Conclusions. The height of the elimination tree has long acted as the only criterion in deriving a suitable fill-preserving sparse matrix ordering for parallel factorization. Although it is well known that adopting height as the criterion for all circumstances is deficient, this has never been successfully addressed by any research. This study is the first one to expand the ordering criterion into a more general class that reflects the various aspects of parallel factorization. We recognize that if any cost func-

tion satisfies the proposed independent and conservative properties, a greedy ordering scheme then generates an optimal ordering with minimum completion cost.

It should be noted that our emphasis in this paper has been on pursuing optimal equivalent reorderings for parallel Cholesky factorizations from various theoretical perspectives. To justify the results obtained in this study, we still need to conduct a comprehensive experiment on diverse actual parallel Cholesky factorization codes.

Recently, more effective results for parallel factorization have been achieved through a larger task model, such as the clique or supernodal tree [9] [26] [27]. In such a 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. To our knowledge, recent literature [3] [8] only achieved finding an ordering to minimize the height of the clique tree. Methods for finding optimal orderings in terms of other more general and realistic concerns remain unanswered. We are currently engaged in this investigation and expect to have some results in the near future.

Acknowledgements. We would like to thank John G. Lewis for many valuable suggestions and comments, especially on improving the conservation property for the nodal cost function. We thank the referees for many thoughtful suggestions, particularly in pointing out some errors of our previous implementation of the MinCP algorithm. We are also grateful for the support from the National Center for High-performance Computing (NCHC) in using SGI Origin 3800.

REFERENCES

[1] C. Ashcraft and J.W.H. Liu, *Robust ordering of sparse matrices using multisection*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 816–832.

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

[3] J.R.S. Blair and B.W. Peyton, *On finding minimum-diameter clique trees*, Nordic Journal of Computing, 1 (1994), pp. 173–201.

[4] 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.

[5] 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.

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

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

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

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

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

[11] B. Hendrickson and E. Rothberg, *Improving the run time and quality of nested dissection ordering*, SIAM J. Sci. Comput., 20 (1999), pp. 468-489.

[12] 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.

[13] G. Karypis and V. Kumar, *METIS – A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices – Version 4.0,*, University of Minnesota, 1998.

[14] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Sci. Comput., 20 (1999), pp. 359–392.

[15] 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.

[16] 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.

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

[18] W.Y. Lin and C.L. Chen, *On evaluating elimination tree based parallel sparse Cholesky factorizations*, International Journal of Computer Mathematics, 74 (2000), pp. 361-377.

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

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

[21] J.W.H. Liu, *Equivalent sparse matrix reordering by elimination tree rotations*, SIAM J. Sci. Stat. Comput., 9 (1988), pp. 424–444.

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

[23] 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.

[24] F. Pellegrini, J. Roman, and P. Amestoy, *Hybridizing nested dissection and halo approximate minimum degree for efficient sparse matrix ordering*, Concurrency: Practice Experience, 12 (2000), pp. 69–84.

[25] 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.

[26] 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.

[27] E. Rothberg and A. Gupta, *An efficient block-oriented approach to parallel sparse Cholesky factorization*, SIAM J. Sci. Comput., 15 (1994), pp. 1413–1439.

[28] M. Yannakakis, *Computing the minimum fill-in is NP-complete*, SIAM J. Alg. Disc. Methods, 2 (1981), pp. 77–79.

TABLE 4
Test matrices from UF Sparse Matrix Collection.

Key	Description	Order	NZ in $A(10^3)$
3DTUBE	3-D pressure tube	45330	1629
BARTH4	Structural engineering problem	6019	23
BARTH5	Structural engineering problem	15606	61
BCSSTK30	Structural eng., off-shore platform	28924	1036
BCSSTK31	Structural eng., automobile component	35588	608
BCSSTK32	Structural eng., automobile chassis	44609	1029
BCSSTK35	Structural eng., automobile seat frame	30237	740
BCSSTK36	Structural eng., automobile shock absorber	23052	583
BCSSTK37	Structural eng., track ball	25503	583
BCSSTK39	Shuttle solid rocket booster	46772	1068
CFD1	CFD, symmetric pressure matrix	70656	949
CFD2	CFD, symmetric pressure matrix	123440	1605
CRYSTK02	Structural eng., crystal vibration	19365	497
CRYSTK03	Structural eng., crystal vibration	24696	863
CT20STIF	Structural eng., CT20 engine block	52329	1323
FINAN512	Economics, portfolio optimization	74752	336
GEARBOX	ZF aircraft flap actuator	153746	4617
LI	3D FEM, magnetohydrodynamic problem	22695	686
MSC10848	Aircraft structure	10848	620
MSC23052	Naval destroyer structure	23052	588
NASASRB	Shuttle rocket booster	54870	1366
PWT	Structural engineering problem	36519	181
SHUTTLE	Structural engineering problem	10429	57
SKIRT	Structural engineering problem	12598	104
STRUCT3	L-shaped regular grid	53570	613
STRUCT4	Full three ocean model	4350	121
VIBROBOX	Vibroacoustic problem	12328	177

TABLE 5
Completion cost increase (mean) of Jess-Kees to MinCP, initially ordered by MMD.

key	Φ2	Φ3	Φ4	Φ5	Φ6	Φ7	Φ8	Φ9
3DTUBE	+0%	+0%	+1%	+0%	+0%	+0%	+0%	+1%
BARTH4	+10%	+6%	+13%	+5%	+5%	+10%	+6%	+13%
BARTH5	+2%	+1%	+5%	+1%	+1%	+2%	+1%	+5%
BCSSTK30	+16%	+11%	+18%	+4%	+8%	+16%	+11%	+18%
BCSSTK31	+2%	+1%	+5%	+1%	+1%	+2%	+1%	+5%
BCSSTK32	+10%	+6%	+14%	+5%	+7%	+10%	+6%	+14%
BCSSTK35	+11%	+3%	+18%	+2%	+7%	+11%	+3%	+18%
BCSSTK36	+1%	+1%	+4%	+1%	+1%	+1%	+1%	+4%
BCSSTK37	+5%	+1%	+14%	+1%	+6%	+5%	+1%	+14%
BCSSTK39	+6%	+5%	+8%	+2%	+3%	+6%	+5%	+8%
CFD1	+12%	+7%	+18%	+6%	+9%	+12%	+7%	+18%
CFD2	+1%	+1%	+2%	+1%	+1%	+1%	+1%	+2%
CRYSTK02	+1%	+1%	+2%	+1%	+1%	+1%	+1%	+2%
CRYSTK03	+2%	+2%	+3%	+1%	+1%	+2%	+2%	+3%
CT20STIF	+6%	+3%	+14%	+4%	+8%	+6%	+3%	+14%
FINAN512	+0%	+0%	+1%	+0%	+0%	+0%	+0%	+1%
GEARBOX	+1%	+0%	+2%	+0%	+1%	+1%	+0%	+2%
LI	+1%	+0%	+2%	+0%	+1%	+1%	+0%	+2%
MSC10848	+3%	+2%	+3%	+1%	+1%	+3%	+2%	+3%
MSC23052	+1%	+1%	+4%	+0%	+1%	+1%	+1%	+4%
NASASRB	+4%	+4%	+4%	+2%	+1%	+4%	+4%	+4%
PWT	+2%	+5%	+1%	+3%	+0%	+2%	+5%	+1%
SHUTTLE	+3%	+2%	+3%	+1%	+1%	+3%	+3%	+3%
SKIRT	+6%	+7%	+8%	+3%	+2%	+6%	+8%	+8%
STRUCT3	+1%	+1%	+1%	+0%	+0%	+1%	+1%	+1%
STRUCT4	+0%	+0%	+1%	+0%	+0%	+0%	+0%	+1%
VIBROBOX	+0%	+0%	+1%	+0%	+0%	+0%	+0%	+1%

TABLE 6
Completion cost increase (best) of Jess-Kees to MinCP, initially ordered by MMD.

key	Φ2	Φ3	Φ4	Φ5	Φ6	Φ7	Φ8	Φ9
3DTUBE	+11%	+5%	+16%	+4%	+7%	+11%	+5%	+16%
BARTH4	+16%	+12%	+20%	+12%	+14%	+16%	+12%	+20%
BARTH5	+16%	+9%	+29%	+6%	+12%	+16%	+9%	+29%
BCSSTK30	+34%	+26%	+33%	+14%	+16%	+34%	+26%	+33%
BCSSTK31	+22%	+13%	+26%	+11%	+16%	+22%	+13%	+26%
BCSSTK32	+37%	+23%	+39%	+16%	+23%	+37%	+23%	+39%
BCSSTK35	+27%	+9%	+41%	+7%	+21%	+27%	+9%	+41%
BCSSTK36	+18%	+8%	+33%	+9%	+19%	+18%	+8%	+33%
BCSSTK37	+22%	+8%	+36%	+11%	+22%	+22%	+8%	+36%
BCSSTK39	+17%	+15%	+21%	+10%	+10%	+17%	+16%	+21%
CFD1	+27%	+17%	+37%	+12%	+19%	+27%	+17%	+37%
CFD2	+14%	+11%	+20%	+11%	+8%	+14%	+11%	+20%
CRYSTK02	+13%	+9%	+17%	+7%	+12%	+13%	+9%	+17%
CRYSTK03	+21%	+27%	+16%	+16%	+13%	+21%	+27%	+16%
CT20STIF	+27%	+14%	+35%	+16%	+28%	+27%	+14%	+35%
FINAN512	+8%	+2%	+21%	+2%	+10%	+8%	+2%	+21%
GEARBOX	+19%	+10%	+24%	+10%	+16%	+19%	+10%	+24%
LI	+12%	+3%	+23%	+5%	+15%	+12%	+3%	+23%
MSC10848	+22%	+15%	+29%	+13%	+11%	+22%	+15%	+29%
MSC23052	+15%	+10%	+29%	+9%	+20%	+15%	+10%	+29%
NASASRB	+15%	+14%	+17%	+8%	+8%	+15%	+14%	+17%
PWT	+14%	+29%	+9%	+14%	+5%	+14%	+29%	+9%
SHUTTLE	+9%	+10%	+10%	+7%	+6%	+9%	+10%	+10%
SKIRT	+55%	+66%	+37%	+36%	+24%	+55%	+66%	+37%
STRUCT3	+12%	+8%	+18%	+8%	+7%	+12%	+8%	+18%
STRUCT4	+5%	+1%	+15%	+2%	+8%	+5%	+1%	+15%
VIBROBOX	+3%	+1%	+20%	+2%	+6%	+3%	+1%	+20%

TABLE 7
Completion cost increase (mean) of Jess-Kees to MinCP, initially ordered by METIS.

key	Φ2	Φ3	Φ4	Φ5	Φ6	Φ7	Φ8	Φ9
3DTUBE	+0%	+0%	+0%	+0%	+0%	+0%	+0%	+0%
BARTH4	+2%	+0%	+8%	+0%	+2%	+2%	+0%	+8%
BARTH5	+0%	+0%	+3%	+0%	+0%	+0%	+0%	+3%
BCSSTK30	+4%	+0%	+25%	+0%	+4%	+4%	+0%	+25%
BCSSTK31	+2%	+1%	+5%	+1%	+2%	+2%	+1%	+5%
BCSSTK32	+0%	+0%	+2%	+0%	+0%	+0%	+0%	+1%
BCSSTK35	+4%	+1%	+7%	+1%	+4%	+4%	+1%	+7%
BCSSTK36	+1%	+0%	+4%	+0%	+1%	+1%	+0%	+4%
BCSSTK37	+3%	+2%	+7%	+1%	+3%	+3%	+2%	+7%
BCSSTK39	+0%	+0%	+1%	+0%	+0%	+0%	+0%	+1%
CFD1	+3%	+1%	+7%	+1%	+3%	+3%	+1%	+7%
CFD2	+0%	+0%	+0%	+0%	+0%	+0%	+0%	+0%
CRYSTK02	+0%	+0%	+0%	+0%	+0%	+0%	+0%	+0%
CRYSTK03	+0%	+0%	+6%	+0%	+0%	+0%	+0%	+6%
CT20STIF	+1%	+0%	+2%	+0%	+1%	+1%	+0%	+2%
FINAN512	+2%	+2%	+5%	+1%	+2%	+2%	+2%	+5%
GEARBOX	+0%	+0%	+0%	+0%	+0%	+0%	+0%	+0%
LI	+0%	+0%	+5%	+0%	+0%	+0%	+0%	+5%
MSC10848	+1%	+0%	+2%	+0%	+1%	+1%	+0%	+2%
MSC23052	+1%	+0%	+3%	+0%	+1%	+1%	+0%	+3%
NASASRB	+0%	+0%	+1%	+0%	+0%	+0%	+0%	+1%
PWT	+0%	+0%	+2%	+0%	+0%	+0%	+0%	+2%
SHUTTLE	+1%	+1%	+4%	+0%	+2%	+1%	+1%	+4%
SKIRT	+3%	+2%	+7%	+1%	+4%	+3%	+2%	+7%
STRUCT3	+2%	+0%	+8%	+0%	+2%	+2%	+0%	+8%
STRUCT4	+0%	+0%	+1%	+0%	+0%	+0%	+0%	+1%
VIBROBOX	+0%	+0%	+0%	+0%	+0%	+0%	+0%	+0%

TABLE 8
Completion cost increase (best) of Jess-Kees to MinCP, initially ordered by METIS.

key	Φ2	Φ3	Φ4	Φ5	Φ6	Φ7	Φ8	Φ9
3DTUBE	+0%	+0%	+0%	+0%	+0%	+0%	+0%	+0%
BARTH4	+22%	+1%	+61%	+0%	+23%	+22%	+1%	+60%
BARTH5	+5%	+1%	+36%	+0%	+7%	+5%	+1%	+35%
BCSSTK30	+30%	+13%	+50%	+13%	+31%	+30%	+13%	+50%
BCSSTK31	+1%	+1%	+12%	+1%	+0%	+1%	+1%	+12%
BCSSTK32	+15%	+7%	+33%	+10%	+15%	+15%	+7%	+33%
BCSSTK35	+13%	+8%	+21%	+7%	+12%	+13%	+8%	+21%
BCSSTK36	+9%	+9%	+15%	+9%	+12%	+9%	+9%	+15%
BCSSTK37	+23%	+12%	+48%	+12%	+19%	+23%	+12%	+48%
BCSSTK39	+1%	+1%	+21%	+0%	+1%	+1%	+1%	+21%
CFD1	+15%	+11%	+28%	+5%	+17%	+15%	+11%	+28%
CFD2	+0%	+0%	+6%	+0%	+0%	+0%	+0%	+6%
CRYSTK02	+0%	+0%	+0%	+0%	+0%	+0%	+0%	+0%
CRYSTK03	+0%	+0%	+21%	+0%	+0%	+0%	+0%	+21%
CT20STIF	+9%	+2%	+19%	+3%	+10%	+9%	+2%	+19%
FINAN512	+13%	+13%	+13%	+9%	+10%	+13%	+13%	+13%
GEARBOX	+2%	+2%	+9%	+2%	+4%	+2%	+2%	+9%
LI	+0%	+2%	+23%	+0%	+5%	+0%	+2%	+23%
MSC10848	+11%	+3%	+18%	+1%	+9%	+11%	+3%	+18%
MSC23052	+6%	+7%	+15%	+8%	+9%	+6%	+7%	+15%
NASASRB	+0%	+0%	+8%	+2%	+4%	+0%	+0%	+8%
PWT	+9%	+8%	+23%	+6%	+13%	+9%	+8%	+23%
SHUTTLE	+14%	+21%	+26%	+15%	+15%	+14%	+21%	+26%
SKIRT	+13%	+14%	+34%	+10%	+16%	+13%	+14%	+33%
STRUCT3	+14%	+6%	+34%	+3%	+15%	+14%	+6%	+34%
STRUCT4	+4%	+1%	+13%	+0%	+6%	+4%	+1%	+13%
VIBROBOX	+1%	+1%	+1%	+0%	+0%	+1%	+1%	+1%

TABLE 9

Time (CPU seconds) to execute the minimum height (J-K) and minimum completion cost orderings (MinCP).

Key	J-K	Φ2	Φ3	Φ4	Φ5	Φ6	Φ7	Φ8	Φ9
3DTUBE	0.22	0.56	0.55	0.39	0.48	0.51	0.53	0.53	0.52
BARTH4	0.02	0.04	0.04	0.03	0.03	0.04	0.04	0.04	0.04
BARTH5	0.05	0.11	0.11	0.09	0.10	0.10	0.10	0.11	0.11
BCSSTK30	0.08	0.18	0.17	0.13	0.15	0.16	0.17	0.17	0.16
BCSSTK31	0.12	0.28	0.27	0.20	0.24	0.25	0.26	0.27	0.26
BCSSTK32	0.13	0.31	0.30	0.22	0.26	0.28	0.29	0.29	0.28
BCSSTK35	0.08	0.17	0.17	0.12	0.15	0.16	0.16	0.16	0.16
BCSSTK36	0.06	0.13	0.13	0.09	0.11	0.12	0.12	0.12	0.12
BCSSTK37	0.07	0.15	0.15	0.11	0.13	0.14	0.14	0.14	0.14
BCSSTK39	0.14	0.33	0.32	0.24	0.28	0.30	0.31	0.31	0.31
CFD1	0.44	1.22	1.21	0.86	1.05	1.12	1.16	1.16	1.16
CFD2	0.77	2.18	2.23	1.59	2.00	2.08	2.19	2.17	2.18
CRYSTK02	0.06	0.15	0.15	0.11	0.13	0.14	0.14	0.14	0.14
CRYSTK03	0.12	0.29	0.28	0.20	0.25	0.26	0.27	0.27	0.27
CT20STIF	0.17	0.42	0.42	0.31	0.37	0.39	0.41	0.41	0.40
FINAN512	0.34	0.92	1.02	0.71	0.91	0.94	0.94	0.98	0.93
GEARBOX	0.63	1.64	1.64	1.18	1.43	1.53	1.58	1.59	1.52
LI	0.21	0.57	0.56	0.40	0.48	0.51	0.53	0.53	0.52
MSC10848	0.03	0.07	0.07	0.05	0.06	0.06	0.07	0.06	0.06
MSC23052	0.06	0.13	0.13	0.10	0.11	0.12	0.13	0.13	0.13
NASASRB	0.19	0.49	0.48	0.35	0.42	0.45	0.46	0.46	0.46
PWT	0.12	0.31	0.32	0.24	0.28	0.29	0.30	0.31	0.30
SHUTTLE	0.03	0.08	0.08	0.06	0.07	0.07	0.08	0.08	0.08
SKIRT	0.04	0.10	0.09	0.07	0.08	0.09	0.09	0.09	0.09
STRUCT3	0.19	0.50	0.50	0.37	0.44	0.47	0.48	0.49	0.48
STRUCT4	0.04	0.08	0.08	0.06	0.07	0.07	0.08	0.07	0.07
VIBROBOX	0.07	0.16	0.16	0.12	0.14	0.15	0.15	0.15	0.15