International Journal of Management Science
Vol 8, No 1, May 2002

A Minimum Degree Ordering Algorithm using the Lower and Upper Bounds of Degrees[*]

Chan-Kyoo Park[**]
Department of IT Consulting and Auditing, National Computerization Agency,
Suji-Eub, Yongin-Si, Kyonggi-do, 447-717, Korea

Seungyong Doh, Soondal Park[***]
Department of Industrial Engineering, Seoul National University,
Shinlim-dong, Kwanak-gu, Seoul, 151-742, Korea

Woo-Je Kim[****]
Department of Industrial Engineering, Daejin University
Pochun-gun, Kyonggi-do, 487-800, Korea

(Received Feb. 2001, Revised Jan. 2002, Accepted Mar. 2002)

ABSTRACT

Ordering is used to reduce the amount of fill-ins in the Cholesky factor of a symmetric positive definite matrix. One of the most efficient ordering methods is the minimum degree ordering algorithm(MDO). In this paper, we provide a few techniques that improve the performance of MDO implemented with the clique storage scheme.
First, the absorption of nodes in the cliques is developed which reduces the number of cliques and the amount of storage space required for MDO. Second, we present a modified minimum degree ordering algorithm of which the number of degree updates can be reduced by introducing the lower bounds of degrees. Third, using both the lower and upper bounds of degrees, we develop an approximate minimum degree ordering algorithm. Experimental results show that the proposed algorithm is competitive with the minimum degree ordering algorithm that uses quotient graphs from the points of the ordering time and the nonzeros in the Cholesky factor.

[*] Supported by Korea Science and Engineering Foundation(KOSEF) under Grant No. 2000-1-31500-001-02.
[**] Email: parkck@nca.or.kr
[***] Email: {dsy, sdpark}@orlab.snu.ac.kr
[****] Email: wjkim@road.daejin.ac.kr

1. INTRODUCTION

Interior point methods for a linear programming problem compute a direction vector at each iteration. Most of the computations at each iteration are involved in the solution of a large sparse linear system, $A\Theta A^T x = b$, where Θ is a diagonal matrix ([10]). The linear systems are typically solved by factoring $M = A\Theta A^T$ into $M = LL^T$, where L is a lower triangular matrix with positive diagonal elements. The matrix L is called the *Cholesky factor* of M.

To factorize large sparse positive matrices into Cholesky factor efficiently, it is necessary to keep the sparsity of L as high as possible because the total computations required are determined by the number of nonzeros of L. On the other hand, the sparsity of L depends heavily on the sequence of the rows of M. So, the number of nonzeros of L can be reduced by ordering of the rows (or columns) of M.

Finding the optimal ordering that minimizes the number of nonzeros of L is known to be NP-complete ([18]). Heuristic methods for carrying this out are minimum degree ordering, minimum deficiency ordering, and nested dissection ([8, 9, 11, 15, 17]). Of the three heuristic methods, the minimum degree ordering algorithm (MDO) is the most widely used because of its good computational performance.

MDO was first suggested by Tinney and Walker [17]. Subsequently, various enhancement techniques for MDO have been developed: indistinguishable nodes, incomplete update, multiple elimination, and external degree ([8, 12]). An approximate minimum degree ordering algorithm was recently developed by Amestoy et al. [1].

For the fast implementation of MDO, the elimination graphs need to be updated efficiently. There are two schemes for storing the elimination graphs: quotient graph scheme and clique storage scheme. Quotient graphs were suggested by George and Liu [6], and most of the ordering codes use quotient graphs ([6, 7]). In the quotient graph scheme, the adjacent nodes of a node is calculated from the adjacent list of the node. On the other hand, the clique storage scheme was suggested by Speelpenning [16] before quotient graphs. In the clique storage scheme, the elimination graph is split into the cliques throughout ordering, and the adjacent nodes of a node are calculated by searching all cliques including the node. Recently, a hybrid scheme was proposed where some of the adjacency structure of the elimination graphs are expressed with quotient graphs, and the others are expressed with cliques ([1]). However, no efficient implementation of MDO that uses the clique storage scheme are known to the authors. Hence, in this paper we provide a few techniques that can improve the performance of MDO using the clique storage scheme.

The organization of the paper is as the following: First, the absorption of nodes in the cliques is developed which reduces the number of cliques and the amount of storage space required for MDO (in Section 2). Second, we present a modified minimum degree ordering algorithm of which the number of degree updates can be reduced by introducing of the lower bounds of degrees (in Section 3). Third, using both the lower and the upper bounds of degrees, we develop an approximate minimum degree ordering algorithm (in Section 4). Finally, Computational results show that the proposed algorithm is competitive with the minimum degree ordering algorithm that uses quotient graphs from the points of the ordering time and the nonzeros in the Cholesky factor (in Section 5).

2. THE MINIMUM DEGREE ORDERING USING THE CLIQUE STORAGE SCHEME

MDO is a symmetric version of Markowitz's ordering. It selects the next nodes of minimum degree from the elimination graphs. For details, see [7].

Let $M = A\Theta A^T$ be an $m \times m$ symmetric positive definite matrix, where A is an $m \times n$ matrix and Θ is a diagonal matrix with positive diagonal values. The nonzero pattern of M can be expressed by the graph $G = (N, E)$ where $N = \{1,, \cdots, m\}$ denotes the set of rows (columns). An undirected edge (i,j) is in E if and only if $a_{ij} \neq 0,\ i \neq j$. We call the graph G the *associated graph* of M. Let $Adj_G(i)$ denote the set of nodes adjacent to i in G, that is, $Adj_G(i) = \{u \in N \mid (u,i) \in E\}$, and $Deg_G(i)$ denotes the degree of i. Note that $Deg_G(i) = |Adj_G(i)$ where $|\cdot|$ is the cardinality of a set.

At the (i-1)-th step, MDO selects node x of minimum degree from $G_{i-1} = (N_{i-1}, E_{i-1})$. Then, the selected node x is eliminated from G_{i-1} and some edges are added. Formally, the transformed graph $G_i = (N_i, E_i)$ from G_{i-1} is obtained as the following:

$$\begin{aligned} N_i &\leftarrow N_{i-1} - \{x\} \\ E_i &\leftarrow (E_{i-1} - \{(u,x) \mid (u,x) \in E_{i-1}\}) \cup \\ &\quad \{(v,y) \notin E_{i-1} \mid (x,v) \in E_{i-1}, (x,y) \in E_{i-1}\} \end{aligned}$$

MDO repeats the same procedure until all of the nodes are eliminated. At the first step, we set G_0 to G. The series of graphs, $G_0, G_1, \cdots$, are called the *elimination graphs*.

Most of the computations in MDO are involved in transforming graph G_{i-1} to G_i and updating the degrees of nodes. Consequently, the storage scheme for the elimination graphs greatly affects the performance of MDO. There are two storage schemes: the quotient graph model and the clique storage scheme. In this study, we consider only the clique storage scheme. For the quotient graph model, see [6] and [7].

In the clique storage scheme, the elimination graph G_i is split into cliques, and transformation of the elimination graphs corresponds to merging some cliques. In graph theory, a clique C is defined as a graph in which there is an edge between every pair of two distinct nodes. Let $n(C)$ and $e(C)$ denote the set of nodes and the set of edges, respectively, of the clique C. The number of nodes of the clique C is called the *clique size of C*. Let $C(V)$ denote the clique whose node set is V. Let K denote a set of cliques $\{C_1, C_2, \cdots, C_r\}$. If the following two conditions are satisfied, K is called a *clique cover* of $G = (N, E)$.

$$n(C_1) \cup n(C_2) \cdots \cup n(C_r) = N$$
$$e(C_1) \cup e(C_2) \cdots \cup e(C_r) = E\ .$$

The cardinality of K, $|K|$, is called the *size of clique cover K*.

Let K be a clique cover of G. Also, let $\{C_1^u, \cdots, C_k^u\}$ be the subset of K that consists of all the cliques including the node u. Then, the set of nodes adjacent to u is represented by the following:

$$Adj_G(u) = n(C_1^u) \cup \cdots \cup n(C_k^u) - \{u\}\ .$$

Let node x be selected as the next node to be deleted from G_{i-1}. Also, let K_{i-1} denote a clique cover of G_{i-1}. Let $C_1^x, \cdots, C_k^x$ denote the cliques whose node sets include the node x. Transforming from G_{i-1} to G_i corresponds to updating the clique cover K_{i-1} of G_{i-1} into the clique cover K_i of G_i. Let $C(V)$ denote the clique with $V = n(C_1^x) \cup \cdots \cup n(C_k^x) - \{x\}$. Then, K_i can be obtained as the following:

$$K_i = (K_{i-1} - \{C_1^x, \cdots, C_k^x\}) \cup \{C(V)\}\ .$$

We call this procedure *clique merging*. It merges all the cliques including node x into the new clique $C(V)$. We call $C(V)$ the *merged clique* of $C_1^x,\cdots,C_k^x$. After merging the cliques, the degree of nodes adjacent to x in G_{i-1} need to be updated to the degrees in G_i. Merging the cliques and updating the degrees spend most of the execution time of MDO.

In interior point methods for linear programming, MDO determines the order of the rows (columns) of matrix $M = A\Theta A^T$. Throughout this paper, it is assumed, without loss of generality, that each row and column of A have at least one nonzero element. Also, no accidental cancellation is assumed in matrix multiplication and addition.

Let $G=(N,E)$ be the associated graph of matrix M. For any subset V of N, the graph $G(V)=(V, E(V))$, where $E(V)=\{(i,j)\in E \mid i\in V, j\in V\}$ and is called a *induced subgraph* of G. Let A_s denote the s-th column of matrix A and $V=\{r_1, r_2,\cdots, r_l\}$ be the set of row indices of nonzero elements of A_s. It is known that that the induced subgraph $G(V)$ of G is a clique ([13]). Moreover, $K=\{C_1,\cdots,C_n\}$ is a clique cover of G where $C_i (i=1,\cdots,n)$ is the clique whose node set is equal to the set of row indices of nonzero elements of A_i.

There may exist many clique covers of the graph G. However, the clique covers of smaller size are better with respect to execution time and storage requirement. Mostly, the computation time for transforming the elimination graphs and updating degrees is expected to be proportional to the number of cliques involved in these operations. Also, we may expect that the clique cover of smaller size needs less storage space. Finding a clique cover of G with minimum size is known to be NP-complete ([4]). One of the techniques for obtaining a clique cover of smaller size is element absorption ([2]).

Let K be a clique cover of G with $K=\{C_1,C_2,\cdots,C_r\}$. The basic idea of element absorption is that if $n(C_i)\subset n(C_j)$ for some i and j, then $K'=K-\{C_i\}$ is also a clique cover of G with $|K'|=|K|-1$. In practice, finding a clique that is absorbed by other cliques may require much computational effort. In the next theorem, the generalization of element absorption is provided, which enables us to perform element absorptions with a small amount of computation.

Theorem 1: Let K denote a clique cover of G, and $C_1^u,\cdots,C_k^u$ denote all the cliques of K that include node u. Suppose that for a clique $C_j^u (1\le j\le k)$, there exists $I\subset\{1,2,\cdots,k\}-\{j\}$ such that

$$e(C_j^u) \subset \bigcup_{i \in I} e(C_i^u) .$$

Let V be $n(C_j^u) - \{u\}$ and K' denote

$$(K - \{C_j^u\}) \cup \{C(V)\} .$$

Then, K' is also a clique cover of G.

Proof: If $n(C_j^u) = \{u\}$, K' is obviously a clique cover of G. Suppose that $n(C_j^u) \neq \{u\}$. Let v be any node in $n(C_j^u) - \{u\}$. Then, there exists the edge (v,u) in G. It is enough to show that the edge set of at least one clique, except C_j^u, includes the edge (v, u). If the condition of the theorem is satisfied, then $e(C_j^u) \subset \bigcup_{i \in I} e(C_i^u)$ and $v \in \bigcup_{i \in I} n(C_i^u)$. That is, there exists a clique C_t^u, such that $t \in I$ and $v \in C_t^u$. By the definition of I, the clique C_t^u, includes the edge (u,v). □

By Theorem 1, we reduce the number of cliques that include node u. We call this procedure in Theorem 1 *node absorption*. Every element absorption can be accomplished by performing node absorptions repeatedly. If the cliques of a clique cover of the elimination graph is sorted in non-increasing order of their clique sizes, then node absorptions can be performed with a little computation during the calculation of the set of adjacent nodes. To perform element absorption, $(|K|(|K|-1)/2)$ additional comparisons among the cliques are required, while node absorption can be performed by only searching the cliques including adjacent nodes during updating the degrees of nodes.

3. A minimum degree ordering using the lower bounds of degrees

By representing the elimination graphs by the clique covers, we easily obtain the lower bounds and the upper bounds of degrees of nodes.

Theorem 2: Let K be a clique cover of G, and $C_1^u, \cdots, C_k^u$ be all the cliques of K that include node u. Then,

$$\max_{1 \le i \le k} (|n(C_i^u)| - 1) \le Deg_G(u) \le \sum_{i=1}^{k} (|n(C_i^u)| - 1)$$

Proof: Since $(n(C_i^u) - \{u\})$ is a subset of $Adj_G(u)$ for all $i (1 \le i \le k)$, we find

$$Deg_G(u) = |Adj_G(u)| \ge |n(C_i^u) - \{u\}|.$$

Therefore, we obtain a lower bound of the degree of u as the following:

$$Deg_G(u) \ge \max_{1 \le i \le k} (|n(C_i^u)| - 1).$$

Also, we can obtain an upper bound of the degree of u as the following:

$$\begin{aligned} Deg_G(u) &= |Adj_G(u)| = |(n(C_1^u) - \{u\}) \cup \cdots \cup (n(C_k^u) - \{u\})| \\ &\le \sum_{i=1}^{k} (|n(C_k^u)| - 1). \end{aligned}$$

□

Two nodes, u and v, are said to be *indistinguishable* if

$$Adj_G(u) \cup \{u\} = Adj_G(v) \cup \{v\}$$

If u and v are indistinguishable in the elimination graph G_{i-1}, these nodes remain indistinguishable until they are deleted from the elimination graphs. Therefore, to treat indistinguishable nodes as one supernode speeds up the performance of MDO. The next lemma provides lower bounds of degrees for the case where the indistinguishable nodes are deleted simultaneously.

Lemma 3: Suppose that x_1 is selected as the next node to be deleted from G_{i-1}. Let $\{x_1, x_2, \cdots, x_p\}$ denote the set of indistinguishable nodes from x_1 in the elimination graph G_{i-1}. If the indistinguishable nodes are deleted simultaneously, then the degrees of nodes in G_i transformed from G_{i-1} satisfy the following inequality:

$$Deg_{G_i}(u) \ge Deg_{G_{i-1}}(u) - p, \quad \forall u \in N_i.$$

Proof: If u is not an element of $Adj_{G_{i-1}}(x_1)$, then $Deg_{G_i}(u)$ is equal to $Deg_{G_{i-1}}(u)$. Therefore, the lemma is satisfied. Next, suppose that u is an element of $Adj_{G_{i-1}}(x_1)$. Then, we find that

$$\begin{aligned}
|Deg_{G_i}(u)| &= |Adj_{G_i}(u)| \\
&= |Adj_{G_{i-1}}(u) \cup (Adj_{G_{i-1}}(x_1) - \{u\}) - \{x_1, \cdots, x_p\}| \\
&= |Adj_{G_{i-1}}(u) \cup (Adj_{G_{i-1}}(x_1) - \{u\})| - |\{x_1, \cdots, x_p\}| \\
&\geq |Adj_{G_{i-1}}(u)| - p = Deg_{G_{i-1}}(u) - p.
\end{aligned}$$

□

Lemma 3, which considers the case where the indistinguishable nodes are deleted simultaneously, is a generalized version of observation 2 in [8]. Another lower bound of the degree can be derived from the clique covers of the elimination graphs. Let K_i denote a clique cover of elimination graph G_i, and K_i^u denote the subset of K_i that consists of all the cliques including node u. Let $Ldeg_{G_0}(u) = Deg_{G_0}(u)$ for all $u \in N_0$. In the elimination graph G_i $(i \geq 1)$, we define another lower bound of the degree, $Ldeg_{G_i}(u)$, recursively as the following:

$$Ldeg_{G_i}(u) = \begin{cases} \max\{(Ldeg_{G_{i-1}}(u) - p), (\max_{C \in K_i^u} |n(C)| - 1)\} & \text{if } u \in Adj_{G_{i-1}}(x_1) \\ Ldeg_{G_{i-1}}(u) & \text{if } u \notin Adj_{G_{i-1}}(x_1). \end{cases}$$

Theorem 4: For any i, the following inequality is satisfied:

$$Ldeg_{G_i}(u) \leq Deg_{G_i}(u), \quad \forall u \in N_i.$$

Proof: By Theorem 2 and Lemma 3, the theorem obviously holds. □

It is possible to avoid updating degrees of nodes, which may not be the minimum degree nodes of the next elimination graphs, by using lower bounds of degrees. The minimum degree ordering algorithm using lower bounds of degrees (MDOL) is as follows:

A minimum degree ordering algorithm using the lower bounds of degrees (MDOL)

```
1:  Set $G_0 \leftarrow G = (N, E)$ and $i = 0$.
2:  Calculate $Deg_{G_i}(u)$ for all $u \in N$.
3:  Set $L \leftarrow \emptyset, D \leftarrow N$, and $S \leftarrow \emptyset$.
4:  While $N - S \neq \emptyset$ do
5:      Compute $mindeg = \min_{j \in D} Deg_{G_i}(j)$ .
6:      For $u \in \{v \in L \mid Ldeg_{G_i}(v) \leq mindeg\}$,
7:          compute $Deg_{G_i}(u)$ and
8:          set $L \leftarrow L - \{u\}$ and $D \leftarrow D \cup \{u\}$.
9:      Find $x_1$ such that $Deg_{G_i}(x_1) = \min_{j \in D} Deg_{G_i}(j)$.
10:     Set $S \leftarrow S \cup \{x_1, x_2, \cdots, x_p\}$.
11:     Transform $G_i$ to $G_{i+1}$ .
12:     For each $u \in Adj_{G_i}(x_1)$,
13:         calculate $Ldeg_{G_{i+1}}(u)$ and
14:         set $L \leftarrow L \cup \{u\}$ and $D \leftarrow D - \{u\}$.
15:         Set $i \leftarrow i + 1$.
16:     End of While
```

In MDOL, L represents the set of nodes for which only the lower bounds of the degrees are calculated. On the other hand, D is the set of nodes of which the degrees are calculated. The set S is the set of nodes that has been deleted from the elimination graphs.

MDOL is different from the minimum degree ordering algorithm in two ways. First, MDOL does not update the degrees of the nodes that are adjacent to the deleted nodes. Instead, MDOL updates only the lower bounds of the degrees of those nodes. Second, MDOL calculates the degrees of nodes that are only expected to have minimum degree. That is, the calculation of degrees is considered only for the nodes for which the lower bounds of degrees are less than or equal to the upper bound of the minimum degree of the elimination graphs. If the nodes that are adjacent to the deleted nodes have large degrees, MDOL can avoid unnecessary degree updates of those nodes.

4. A MINIMUM DEGREE ORDERING ALGORITHM USING THE LOWER AND UPPER BOUNDS OF DEGREES

Let $G_i = (N_i, E_i)$ denote the elimination graph at the i-th iteration of MDO. For any node $v \in N_i$, let K_i^v denote the subset of the clique cover K_i of G_i, where all of the cliques in K_i^v include node v. Also, for any subset X of N_i, let K_i^X denote the subset of K_i where $K_i^X = \bigcup_{x \in X} K_i^x$.

Lemma 5: Suppose $X = \{x_1, \cdots, x_p\}$ is a set of indistinguishable nodes and the nodes of X are selected as the nodes to be deleted from G_{i-1}. Also, let C_* denote the merged clique of all the cliques of K_{i-1}^X. For any $v \in Adj_{G_{i-1}}(x_1)$,

$$Deg_{G_i}(v) \le Deg_{G_{i-1}}(v) + |\, n(C_*) \,| - \max_{C \in K_{i-1}^v \cap K_{i-1}^X} |\, n(C) \,| .$$

Proof: Let $Deg_{G_i}(v) = Deg_{G_{i-1}}(v) + \Delta$ for some integer Δ. And let V denote the set of adjacent nodes of v in the induced subgraph $G_{i-1}(Adj_{G_{i-1}}(x_1) - X)$. The newly added edges to G_i are only the edges whose both endpoints are in $Adj_{G_{i-1}}(x_1) - X$ and which are not in $G_{i-1}(Adj_{G_{i-1}}(x_1) - X)$. Therefore, the number of nodes newly adjacent to node v in G_i is $|\, Adj_{G_{i-1}}(x_1) - X \,| - 1 - |\, V \,|$. Since all nodes of X are deleted from G_{i-1} simultaneously, p edges are removed. It follows that

$$\Delta = |\, Adj_{G_{i-1}}(x_1) - X \,| - 1 - |\, V \,| - p.$$

Let the set of the removed edges be P. Here, $|\, Adj_{G_{i-1}}(x_1) - X \,| = |\, n(C_*) \,|$. And, $|\, V \,| + p \ge \max_{C \in K_{i-1}^v \cap K_{i-1}^X} |\, n(C) \,| - 1$, since any clique $C \in K_{i-1}^v \cap K_{i-1}^X$ is composed of all edges in P and a subset of V. Thus, we obtain

$$\Delta \le |\, n(C_*) \,| - \max_{C \in K_{i-1}^v \cap K_{i-1}^X} |\, n(C) \,| .$$

□

Let $Udeg_{G_0}(v) = Deg_{G_0}(v)$ for all $v \in N_0$. In the elimination graph $G_i (i \geq 1)$, we define the upper bound of the degree, $Udeg_{G_i}(v)$, recursively as the following:

$$Udeg_{G_i}(v) = \begin{cases} \min(Udeg_{G_{i-1}}(v) + |n(C_*)| - |n(C^v_{\max})|, & \\ \quad (\sum_{C \in K^v_i} |n(C)| - 1)\} & \text{if } v \in Adj_{G_{i-1}}(x_1) \\ Udeg_{G_{i-1}}(v) & \text{if } v \notin Adj_{G_{i-1}}(x_1). \end{cases}$$

where C_* and $C^v_{\max}$ are defined in Lemma 5.

Theorem 6: For any i, the following inequality is satisfied:

$$Udeg_{G_i}(v) \geq Deg_{G_i}(v), \quad \forall v \in N_i.$$

Proof: The theorem is derived directly from Theorem 2 and Lemma 5. □

Now that the lower and upper bound of the degree of each node can be obtained by Theorem 4 and 6, we can calculate an approximate degree of each node. An approximate degree of a node v can be set by the function $f(Ldeg_{G_i}(v), Udeg_{G_i}(v))$, which must satisfy the following inequality:

$$Ldeg_{G_i}(v) \leq f(Ldeg_{G_i}(v), Udeg_{G_i}(v)) \leq Udeg_{G_i}(v), \quad \forall v \in N_i.$$

Although the degree of any node can be approximated, the approximation are only applied to the nodes for which the gaps between the lower and upper bounds are small, in order to keep the differences between the approximate degrees and the exact degrees remaining moderately small. An approximate minimum degree ordering algorithm (MDOLU) that calculates the approximate degrees by lower and upper bounds of degrees is as follows:

An approximate minimum degree ordering algorithm using the lower and upper bounds of degrees (MDOLU)

```
1: Set $G_0 \leftarrow G = (N, E)$ and $i = 0$.
2: Calculate $Deg_{G_i}(v)$ for all $v \in N$.
3: Set $Adeg_{G_i}(v) \leftarrow Deg_{G_i}(v)$ for all $v \in N$.
```

```
4:  Set $L \leftarrow \emptyset, D \leftarrow N$, and $S \leftarrow \emptyset$.
5:  While $N - S \neq \emptyset$ do
6:      Compute $minadeg = \min_{j \in D} Adeg_{G_i}(j)$ .
7:      For $u \in \{v \in L \mid Ledg_{G_i}(v) \leq minadeg\}$,
8:          if $Udeg_{G_i}(u) - Ldeg_{G_i}(u) \leq \delta$ , then
9:              set $Adeg_{G_i}(u) \leftarrow f(Ldeg_{G_i}(u), Udeg_{G_i}(u))$.
10:         else
11:             compute $Deg_{G_i}(u)$ and set $Adeg_{G_i}(u) \leftarrow Deg_{G_i}(u)$.
12:         set $L \leftarrow L - \{u\}$ and $D \leftarrow D \cup \{u\}$.
13:     Find $x_1$ such that $Adeg_{G_i}(x_1) = \min_{j \in D} Adeg_{G_i}(j)$.
14:     Set $S \leftarrow S \cup \{ x_1, x_2, \cdots, x_p \}$ where $x_1, \cdots, x_p$ are indistinguishable.
15:     Transform $G_{i-1}$ to $G_i$ .
16:     For each $u \in Adj_{G_{i-1}}(x_1)$.
17:         calculate $Ldeg_{G_i}(u)$ and $Udeg_{G_i}(u)$ and
18:         set $L \leftarrow L \cup \{u\}$ and $D \leftarrow D - \{u\}$.
19:     Set $i \leftarrow i + 1$.
20: End of While
```

In MDOLU, $Adeg_{G_i}(v)$ has the approximate degree or the degree of v in the i-th elimination graph G_i. The set D of nodes includes the nodes for which the degrees or the approximate degrees in G_i are known. On the other hand, the set L of nodes includes the nodes for which only the lower and upper bounds of degrees are calculated. In MDOLU, when some nodes are selected as the next deleted nodes in G_i, the approximate degrees or the degrees of those nodes in G_i are calculated. On the other hand, in MDOL the degrees of the nodes that are selected as the next deleted nodes have to be calculated. This implies that the number of degree updates of MDOLU is usually less than that of MDOL. Note that if δ is set to 0 in MDOLU, MDOLU is the same with MDOL.

5. IMPLEMENTATION AND COMPUTATIONAL RESULTS

Before the computational results are presented, some important implementation techniques for improving the performance of MDOL and MDOLU are discussed.

The technique used for detecting indistinguishable nodes has much effect on the computational time of MDO. For any node u, we define $Nbd_{G_i}(u)$ as $Adj_{G_i}(u) \cup \{u\}$. A new hashing technique for detecting indistinguishable nodes is proposed as follows: First, the integer value $H(u)$ is assigned to each node u before the ordering is started. During the calculation of the exact degree of node u, the following sum $S_{G_i}(u)$ is also computed:

$$S_{G_i}(u) = \left(\sum_{j \in Nbd_{G_i}(u)} H(j) \right) \quad mod \quad MAXINT.$$

where $MAXINT$ is a large positive integer. If the degree of node u is equal to the degree of node v and $S_{G_i}(u)$ is equal to $S_{G_i}(v)$, then the two nodes u and v are regarded as indistinguishable. In our implementation, we set $H(u)$ as the following:

$$H(u) = \lfloor \log(c_1 \times u + c_2) \rfloor + \lfloor (\log(c_1 \times u + c_2) - \lfloor \log(c_1 \times u + c_2) \rfloor) * 10^7 \rfloor$$

where c_1 and c_2 are positive integers. However, this indistinguishable node detection technique may mistake 'not' indistinguishable nodes for indistinguishable nodes. From our experimental results of various linear programming problems, the possibility of the mistakes seems very small.

MDOLU calculates the approximate degrees of nodes using the lower and upper bounds of degrees. In our implementation, the approximation of the degree of any node u is allowed if the difference between the lower and upper bounds of degrees is less than or equal to δ. The appropriate value of δ depends on the number of rows of matrix A. For the case where the number of rows of A is less than 500, δ is set to 10. If the number of rows of A is greater than 1000, δ is set to 20, otherwise, δ is set to 15. Given the lower and upper bound of the degree of node u, the approximate degree of u, $f(Ldeg_{G_i}(u), Udeg_{G_i}(u))$, is calculated by the following:

$$f(Ldeg_{G_i}(u),\ Udeg_{G_i}(u)) = Ldeg_{G_i}(u) + 0.6 \times (Udeg_{G_i}(u) - Ldeg_{G_i}(u)).$$

The equation above is derived from our experiments that the exact degrees tend to be slightly greater than the mean value of the lower and upper bounds of degrees.

We also use the multiple elimination technique, external degrees of nodes, and node absorption. The approximate external degree of a node is set to the approximate degree of the node minus the number of indistinguishable nodes detected.

As more nodes are deleted, the elimination graphs become more dense. This implies that the corresponding submatrix of Cholesky factor L becomes more dense. Therefore, at some stage of the ordering procedure, dense window detection will be desirable. In our implementation, if the minimum of the upper bounds of degrees is sufficiently close to the number of nodes remaining in the elimination graphs, the ordering procedure will terminate immediately after deleting the remaining nodes all together.

The experimental results are provided in Tables 1. The computational experiments were carried out on a SunSparc Ultra 170 (128M). The first four columns of Table 1 describe the name of the problem, the number of rows, the number of columns, and the number of nonzeros of matrix A. To compare ordering times and the number of nonzeros of the Cholesky factor effectively, only large problems are selected from NETLIB ([5]) and the University of Iowa.[1] The fifth column represents the ordering time of MDO, which uses multiple elimination and external degree, but none of lower and upper bounds of degrees. The sixth column represents the number of nonzeros of Cholesky factor L, and the seventh column NUPD represents the total number of degree updates, that is, the number of calculations of exact degree except the initializations of the degrees of nodes, during MDO. The eighth column represents the total storage space required for the initial clique cover after node absorptions are applied. If node absorption is not carried out, then the amount of the storage space required will be equal to the number of nonzeros of A. The ninth ~ eleventh columns represent the ordering time, the number of nonzeros of L, and the number of degree updates of MDOL, respectively. Similarly, the twelfth ~ fourteenth columns represent the ordering time, the number of nonzeros of L, and the number of degree updates of MDOLU, respectively. In addition, Table 1 also shows the computational results of two other ordering codes: Liu's MMD code [12], and CPLEX (ver 4.0) Barrier Solver's approximate minimum degree ordering routine (CPLEXAMD) in the fifteenth ~ eighteenth columns.

[1] ftp://col.biz.uniowa.edu/pub/testprob/lp

Table 1. Experimental results

Problem				MDO				MDOL			MDOLU			MMD		CPLEXAMD	
Name	Rows	Cols	Nonz	Time	Nonz	NUPD	Storage	Time	Nonz.	NUPD	Time	Nonz.	NUPD	Time	Nonz.	Time	NonZ.
df1001	6071	12230	35632	19.05	1601797	102762	33486	2.55	1624748	20723	2.30	1508096	13635	4.01	1634257	-	-
pilot8	2030	6460	72479	11.49	442100	49911	32750	0.79	425414	5597	0.78	449504	4126	2.95	421194	0.70	458132
d2q06c	2171	5831	33081	0.57	137059	13520	15726	0.21	152341	3817	0.23	164947	2549	0.21	165676	0.12	155279
truss	1000	8806	27836	0.23	52300	5938	13890	0.16	58001	2710	0.17	58195	2422	0.05	52509	0.05	60311
bnl2	2280	4442	14952	0.35	84329	11840	11968	0.15	78949	4500	0.16	88902	2137	0.15	82185	0.09	93318
woodw	1098	8418	37487	0.47	48151	8002	19224	0.14	47722	1832	0.15	47693	1466	0.15	47657	0.13	54346
greenbea	2389	5495	30908	0.63	76239	12699	29037	0.18	77837	2909	0.20	82633	1765	0.56	81914	0.20	90330
co5	5715	12125	57308	3.03	172319	29181	41252	0.51	173781	11245	0.43	181483	3589	2.61	177705	0.17	211975
co9	10694	22527	107757	8.88	472480	66331	79230	1.19	477392	24354	1.05	484987	8429	9.93	488456	2.15	556481
cq9	9247	21187	95117	6.84	421681	51691	70822	1.02	411254	20267	0.91	425425	7526	8.40	421178	1.84	496570
nl	7031	15290	46808	3.25	277897	41302	36871	0.70	278620	16668	0.62	278046	5398	5.38	272462	0.67	323739
cre-b	7240	77137	260785	18.03	946555	63855	224491	2.16	947192	11637	2.15	936117	8940	7.60	940374	2.53	1000243
ken-13	28632	42659	97246	5.01	317779	74395	97175	1.93	339245	45678	1.43	317505	11265	5.35	272462	2.03	422835
osa-07	1118	25067	144812	0.78	54807	204	75912	0.43	54807	49	0.37	54789	5	3.76	54783	0.34	62314
osa-14	2337	54797	317097	1.96	116183	228	165795	1.07	116183	49	0.84	116165	5	15.88	116160	1.26	126395
p10	10081	19081	117991	2.06	528559	56212	116910	1.15	492455	36518	0.91	513556	12304	5.81	505060	1.69	638571
r05	5171	9671	104126	1.29	475232	27881	103455	0.80	460212	18033	0.68	462475	9153	3.14	451545	0.71	560921
lp22	2958	16392	68518	17.55	911305	181618	60819	1.87	900338	14421	1.70	905485	9673	34.88	939768	1.43	1052546
nemsemm2	6922	46579	167433	1.40	113925	19267	40956	0.69	112990	11293	0.55	126491	2034	0.72	121207	0.33	146494
nsir2	4451	10055	154937	5.60	312348	11114	19185	1.04	324800	2577	1.09	317331	1410	11.36	323498	0.89	327043
lpl3	10828	33686	100525	1.73	318510	26842	67729	1.00	320978	18112	0.93	333633	6375	3.61	309287	1.03	382096
mod2	34355	64339	195381	18.68	1832574	240151	155857	3.91	1780305	70259	3.62	1840605	33550	16.24	1768877	4.11	2004754
world	34106	65900	194985	22.81	1717014	249605	154329	3.83	1707765	70482	3.54	1773676	32894	17.33	1725610	4.16	1964142
osa-30	4350	104374	604488	12.06	218534	263	315706	2.19	218534	49	1.59	218520	4	63.05	218511	6.19	231817
bas1lp	5411	9825	587775	243,43	2199939	195285	439248	12.02	2259095	9156	11.60	2166147	7709	22.85	2211385	17.60	2442676
rlfddd	4050	61721	272631	14.82	845926	82928	144551	2.43	838902	19576	2.30	860563	11327	3.98	836732	0.77	915929
route	20894	43019	206782	10.27	3133630	81371	179798	2.79	3263627	44750	2.73	3233627	11583	1.54	3033407	2.05	3200797
rat7a	3136	9408	268908	8.55	2029073	21608	228537	3.21	2071791	5752	3.22	2071751	5717	3.53	1581632	2.31	2325761
nemswrld	6594	27977	191633	13.40	949012	73802	157599	1.60	956292	11472	1.51	907882	8815	5.74	896473	1.37	1018373
model10	4400	16439	149992	2.29	534267	25640	142682	0.63	590322	5871	0.68	509495	4485	1.13	502443	0.50	501783

Comparing the number of nonzeros of A (the fourth column) with the storage space required for the initial clique cover, we observe that node absorptions reduce the amount of the storage space by about 20% ~ 50% of the number of nonzeros of A. However, this reduction of the storage space has only a slight effect on the ordering time in our experiments. This means that it is desirable to use node absorptions to reduce the storage space required during the ordering. MDOL and MDOLU both use node absorptions.

The introduction of lower bounds of degrees dramatically reduces the computational time and the number of degree updates. Compared with MDO, the ordering time of MDOL is reduced to 2 ~ 10 times that of MDO. Also, the number of degree updates of MDOL is reduced to about 1/5 ~ 1/2 times that of MDO. Compared with MDOL, the ordering time and the number of degree updates of MDOLU is reduced by 10% and 50%, respectively, on average. These results imply that using the lower and upper bounds of degrees can reduce significantly the ordering time and the number of degree updates.

MMD code performs the minimum degree ordering algorithm using multiple elimination and external degrees. MMD use the quotient graph data structure. Before calling the ordering routines of MMD, the adjacent list of each node in the initial elimination graph is set up beforehand and passed over to the ordering routines of MMD. The time required to set up the adjacent list of each node is not counted in the ordering times of MMD.

Compared with MMD, MDOLU is more than one and a half times faster than MMD in 25 out of 30 problems. In fact, MDOLU is more than three times faster than MMD in 19 out of 30 problems. For the nonzeros of the Cholesky factor, MDOLU is almost as good as or better than MMD in 8 out of 30 problems. In seventeen problems, the number of nonzeros of the Cholesky factor obtained by MDOLU is within (1 + 0.05) times the number of nonzeros of the Cholesky factor obtained by MMD. In almost all of the 30 problems, it is within (1 + 0.1) times the number of nonzeros of the Cholesky factor obtained by MMD. In only one problem, the number of nonzeros of the Cholesky factor obtained by MODLU is 10% larger than that of the Cholesky factor obtained by MMD. In conclusion, the increase in the number of nonzeros of the Cholesky factor of MDOLU was negligible in almost all tested problems.

Finally, compared with CPLEXAMD, MDOLU is faster than CPLEXAMD in about half of the 30 problems. Moreover, with respect to the number of nonzeros of the Cholesky factor, MDOLU is better than CPLEXAMD. However, since CPLEXAMD is known to use a dense window technique and other techniques to speed up the numerical factorization of the Cholesky factor, the smaller number of nonzeros of the Cholesky factor by MDOLU may not imply directly that the ordering of rows by MDOLU is better than CPLEXAMD for the numerical factorization of the Cholesky factor. Nevertheless, the number of nonzeros of the Cholesky factor is the most widely used criterion for the quality of the ordering.

6. CONCLUSION

In this paper, we presented a minimum ordering algorithm that uses the lower and upper bounds of degrees. By the lower bounds of degrees, unnecessary updates of degree can be delayed. As a result, the number of degree updates of the minimum degree ordering algorithm can be reduced significantly. By the upper bounds of degrees, we suggested another degree approximation technique. Also, the node absorption technique, which is a generalization of element absorption, is proposed. The node absorption reduces the storage space required for the approximate minimum degree ordering algorithm using the clique storage scheme. The node absorption can be performed with a little computational effort.

Finally, the experiment results show that the proposed minimum degree ordering algorithm using the clique storage scheme is comparable to the existing minimum degree ordering algorithm using the quotient graph data structure.

Acknowledgement: We are grateful to anonymous referees for their helpful comments on the first version of this paper.

REFERENCES

[1] Amestoy, P. R., T. A. Davis, I. S. Duff, "An approximate minimum degree ordering algorithm," *SIAM Journal on Matrix Analysis and Applications* 17, 4 (1996), 886-905.

[2] Duff, I. S., J. K. Reid, "'The multifrontal solution of indefinite sparse symmetric linear equations," *ACM Transactions on Mathematical Software*, 9 (1983), 302-235.

[3] Duff, I. S., A. M. Erisman and J. K. Reid, *Direct Methods for Sparse Matrices*, Oxford University Press, 1986.

[4] Garey, M. R., D. S. Johnson, "'Computers and intractability," Bell Telephone Laboratories, Inc., 1979.

[5] Gay, D. M., "Electronic mail distribution for linear programming test problems," *Mathematical Programming Society COAL Newsletter*, 1985.

[6] George, A., J. W. H. Liu, "A fast implementation of the minimum degree algorithm using quotient graphs," *ACM Transactions on Mathematical Software* 6, 3 (1980), 337-358.

[7] George, A., J. W. H. Liu, *Computer solution of large sparse positive definite systems*, Prentice-Hall, 1981.

[8] George, A., J. W. H. Liu, "The evolution of the minimum degree ordering algorithm," *SIAM Review* 31, 1 (1989), 1-19.

[9] Hendrickson, B., E. Rothberg, "Improving the run time and quality of nested dissection ordering," *SIAM Journal on Scientific Computing* 20, 2 (1998), 468-489.

[10] Jung, H. W., R. E. Marsten and M. J. Saltzman, "Numerical factorization methods for interior point algorithms," *ORSA Journal on Computing* 6, 1 (1994), 94-104.

[11] Kim, B. G., M. Seong, S. Park, "'An efficient ordering method and data structure of the interior point method," *Journal of the Korean Opertations Research and Management Science Society* 21, 3 (1996), 63-74.

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

[13] Mo, J., S. Park, "Data structures and the performance improvement of the minimum degree ordering method," *Korea Management Science Review* 12, 2 (1995), 31-42.

[14] Rose, D. J., "A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations," in *Graph Theory and Com-*

puting, R. C. Read, ed., Academic Press, 183-217, 1972.

[15] Seol, T.-R., C.-K. Park, S. Park, "Minimum deficiency ordering with the clique storage structure," *Journal of the Korean Institute of Industrial Engineers* 24, 3 (1998), 407-416.

[16] Speelpenning, B., *"The generalized element method,"* Tech. Rep. UIUCDCS-R-78-946, Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign, IL, 1978.

[17] Tinney, W. F., J. W. Walker, "Direct solutions of sparse network equations by optimally ordered triangular factorization," *Proceedings of the IEEE* 55 (1967), 1801-1809.

[18] Yannakakis, M., "Computing the minimum fill-in is NP-complete," *SIAM J. Algebraic and Discrete Methods* 2 (1981), 77-79.