manuscript No.
(will be inserted by the editor)

Pinar Heggernes · Yngve Villanger

Simple and Efficient Modifications of Elimination Orderings

Abstract One of the most important and well studied problems related to sparse Cholesky factorization is to compute elimination orderings that give as few nonzero entries as possible in the resulting factors. We study the problem of modifying a given elimination ordering through local reorderings. We present new theoretical results on equivalent orderings, including a new characterization of such orderings. Based on these results, we define the notion of k-optimality for an elimination ordering, and we describe how to use this in a practical context to modify a given elimination ordering to obtain less fill. We experiment with different values of k, and report on percentage of fill that is actually reduced from an already good initial ordering, like Minimum Degree.

1 Introduction

When solving large, sparse, positive definite systems of linear equations through Cholesky factorization, one of the crucial concerns is to find good symmetric permutations, also called elimination orderings. In such systems, numerical stability is ensured regardless of the ordering, but the number of nonzero elements in the resulting factors depend highly on the elimination ordering. To save computation time and storage it is thus important to find elimination orderings that give as few nonzero elements as possible. In graph terminology, the problem can be equivalently modeled through the well known *Elimination Game* [15,18], where the input graph G corresponds to the nonzero structure of a given symmetric positive definite matrix A, and the output graph G^+ corresponds to the Cholesky factor of A. The set of edges that is added to G to obtain G^+ is called *fill*, and different output graphs

A preliminary version of this work was presented at PARA 2004.

Department of Informatics, University of Bergen, N-5020 Bergen, Norway. E-mail: pinar.heggernes@ii.uib.no · yngve.villanger@ii.uib.no

with varying fill are produced dependent on the elimination ordering in which the vertices are processed. Computing an ordering that minimizes the number of fill edges is an NP-hard problem [20], and hence various heuristics are proposed and widely used, like Minimum Degree [9], Nested Dissection [7], and combinations of these.

Sometimes, after a good elimination ordering with respect to fill is computed by some heuristic, it is desirable to do some modifications on this ordering in order to achieve better properties for other performance measures, like storage or parallelism, without increasing the size of fill. Elimination orderings that result in the same filled graph are called *equivalent orderings*, and they can be used for this purpose [12]. In this paper, we prove some properties of equivalent orderings, and we give a new characterization of such orderings based on local permutations. These results are useful in practice when one seeks to modify a given ordering in order to group subsets of vertices close to or far from each other without changing fill.

Based on the mentioned results, we move on to orderings that are not equivalent to the given ordering, and we study how local permutations of consecutive vertices in a given elimination ordering can affect the resulting fill. We define the notion of *k-optimality* for an elimination ordering, based on vertex subsets of size k, and we give an algorithm that starts with any given ordering and makes it k-optimal for a chosen k. Finally, we implement a relaxed variant of this algorithm to exhibit its practical potential. For an efficient implementation, we use a data structure suitable for chordal graphs, called *clique trees*. Our tests show that even when we start with a good ordering, like Minimum Degree, substantial amount of fill can be reduced by trying to make the given ordering 3-optimal.

This paper is organized as follows. In the next section, we give a brief background and the necessary notation. Section 3 presents the theoretical results of this paper, and contains a new characterization of equivalent reorderings through swapping consecutive vertices. In Section 4, we define the concept of k-equivalent orderings, and explain how the theoretical results of the previous section can be used to find elimination orderings resulting in less fill. These ideas are demonstrated through implementation details and practical experiments in Section 5, and we conclude in Section 6.

2 Preliminaries

A graph is denoted by $G = (V, E)$, where V is the set of vertices, with $|V| = n$, and $E \subseteq \binom{V}{2}$ is the set of edges, with $|E| = m$. When a graph G is given, we will use $V(G)$ and $E(G)$ to denote the vertices and edges of G respectively. The *neighbors* of a vertex v in G are denoted by the set $N_G(v) = \{u \mid uv \in E\}$, and the *closed neighborhood* of v is $N_G[v] = N_G(v) \cup \{v\}$. For a given vertex subset $A \subseteq V$, $G(A)$ denotes the graph induced by the vertices of A in G. A is a *clique* if $G(A)$ is a complete graph. A vertex v is called *simplicial* if $N_G(v)$ is a clique. The *deficiency* of vertex v in G is $D_G(v) = \{ux \mid u, x \in N_G(v)$ and $ux \notin E\}$, i.e., the set of edges that must be added in order to make $N_G(v)$ a clique; hence $D_G(v)$ is empty if v is simplicial.

A *path* is a sequence of vertices $x_1, x_2, ..., v_x$ such that $x_i x_{i+1}$ is an edge for $1 \leq i < k$. The *length* of a path is the number of edges in it. A *cycle* is a path that starts and ends with the same vertex, and the length of the cycle is the number of vertices or edges it contains. A *chord* of a cycle is an edge connecting two non-consecutive vertices of the cycle. A graph is *chordal* if it contains no induced chordless cycle of length ≥ 4. A graph $H = (V, E \cup F)$ is called a *triangulation* of $G = (V, E)$ if H is chordal.

An *elimination ordering* of G is a function $\alpha : V \leftrightarrow \{1, 2, ..., n\}$. We will say that $\alpha(v)$ is the *position* or *number* of v in α, and also use the equivalent notation $\alpha = (v_1, v_2, ..., v_n)$ meaning that $\alpha(v_i) = i$. For the opposite direction we use $\alpha^{-1}(i) = v_i$ to find a vertex, if the number in the ordering is given and $\alpha = (v_1, v_2, ..., v_n)$. The following algorithm simulates the production of fill in Gaussian elimination [15,18].

Algorithm Elimination Game
Input: A graph $G = (V, E)$ and an ordering $\alpha = (v_1, ..., v_n)$ of V.
Output: The filled graph G^+_α.
$G^0_\alpha = G$;
for $i = 1$ **to** n **do**
    $F^i = D_{G^{i-1}_\alpha}(v_i)$;
    Obtain G^i_α by adding the edges in F^i to G^{i-1}_α and removing v_i;
$G^+_\alpha = (V, E + \cup_{i=1}^{n} F^i)$;

We will call G^i_α the i^{th} *elimination graph*. During Elimination Game, two vertices u and v that are not yet processed become *indistinguishable* after step i if $N_{G^i_\alpha}[u] = N_{G^i_\alpha}[v]$, and they remain indistinguishable until one of them is eliminated [8]. Given G and α, we will say that u and v are indistinguishable if u and v are indistinguishable at step $\min\{\alpha(u), \alpha(v)\}$, and otherwise we will say that they are *distinguishable*. The edges that are added to G during this algorithm are called *fill edges*, and uv is an edge in G^+_α if and only if $uv \in E$ or there exists a path $u, x_1, x_2, ..., x_k, v$ in G where $\alpha(x_i) < \min\{\alpha(u), \alpha(v)\}$, for $1 \leq i \leq k$ [17].

If no fill edges are added during Elimination Game, then $G^+_\alpha = G$ and α is called a *perfect elimination ordering* (*peo*). This corresponds to repeatedly removing a simplicial vertex until the graph becomes empty. A graph is chordal if and only if it has a peo [5]; thus filled graphs are exactly the class of chordal graphs, and Elimination Game is a way of producing triangulations of a given graph. Every chordal graph has at least two non-adjacent simplicial vertices [4], and hence any clique of size k in a chordal graph can be ordered consecutively with numbers $n-k+1, n-k+2, ..., n$ in a peo. An elimination ordering is *minimum* if no other ordering can produce fewer fill edges. Ordering α is a *minimal elimination ordering* if no proper subgraph of G^+_α is a triangulation of G, and G^+_α is then called a *minimal triangulation* of G. While it is NP-hard to compute minimum fill [20], minimal fill can be computed in polynomial time [17]. Minimal fill can in general be far from minimum, however algorithms exist to make a given elimination ordering minimal by removing edges from the filled graph [1,3,16]. For our

purposes, when modifying a given elimination ordering to reduce fill, we will always start by making the given ordering minimal by removing the redundant edges in the filled graph using an appropriate existing algorithm. Two elimination orderings α and β are called *equivalent* if $G^+_\alpha = G^+_\beta$ [12]. If α is a minimal elimination ordering then every peo β of G^+_α is equivalent to α [1].

For an efficient implementation of the ideas presented in this paper, we need the notion of clique trees defined for chordal graphs [6]. A *clique tree* of a chordal graph G is a tree T, whose vertex set is the set of maximal cliques of G, that satisfies the following property: for every vertex v in G, the set of maximal cliques containing v induces a connected subtree of T. We will be working on a clique tree of G^+_α to identify groups of vertices that can be locally repermuted to give less fill. Clique trees can be computed in linear time [2]. It follows from the results of [4] that a chordal graph has at most n maximal cliques, and thus a clique tree has $O(n)$ vertices and edges. We refer to the vertices of T as *tree nodes* to distinguish them from the vertices of G. Each tree node of T is thus a maximal clique of G. Clique trees have proved to be useful data structures to represent chordal graphs in order to achieve more efficient algorithms on this graph class [10,11].

3 Equivalent orderings

Equivalent orderings of a given ordering were studied by Liu to achieve better properties for storage without increasing the amount of fill [12]. He used elimination tree rotations to compute appropriate equivalent orderings. Here, we will use repeated swapping of consecutive vertices. First we need the following result from [19].

Lemma 1 [19] *Given G and $\alpha = (x_1, x_2, ..., x_k, u, v, x_{k+3}, ..., x_n)$, let $\beta = (x_1, x_2, ..., x_k, v, u, x_{k+3}, ..., x_n)$ (u and v are swapped). If u and v are non-adjacent in G^+_α then $G^+_\alpha = G^+_\beta$.*

We strengthen this result in the following lemma.

Lemma 2 *Given G and α, let u and v be two vertices in G satisfying $\alpha(u) = \alpha(v) - 1 = i$. Let β be the ordering that is obtained from α by swapping u and v so that $\beta(u) = \alpha(v), \beta(v) = \alpha(u)$, and $\beta(x) = \alpha(x)$ for all vertices $x \neq u, v$. Then $G^+_\alpha = G^+_\beta$ if and only if u and v are non-adjacent in G^+_α or indistinguishable in α.*

Proof (If) If u and v are non-adjacent then it follows from Lemma 1 that $G^+_\alpha = G^+_\beta$. Let us consider the case of indistinguishable vertices. The graph G^+_α is obtained by picking the next vertex v_j in α and then making $N_{G^{j-1}_\alpha}(v_j)$ into a clique and removing v_j from G^{j-1}_α to obtain G^j_α, as described in Elimination Game. This can be considered as making n different sets of vertices into cliques. For every vertex $z \in V \setminus \{u, v\}$, its neighborhood which is made into a clique in G^+_α is the same as its neighborhood which is made into a clique in G^+_β. This follows from Lemma 4 of [17], since the same set of vertices are eliminated prior to z in both orderings. Now we have to show that eliminating u and then v results in the exact same set of fill edges as eliminating v

and then u in the graph $G_\alpha^{i-1} = G_\beta^{i-1}$. Vertices u and v are indistinguishable, so $N_{G_\alpha^{i-1}}[u] = N_{G_\beta^{i-1}}[v]$ which is made into a clique in both cases. Observe that $N_{G_\alpha^i}[v] \subseteq N_{G_\alpha^{i-1}}[u]$ and that $N_{G_\beta^i}[u] \subseteq N_{G_\beta^{i-1}}[v]$, thus making these sets into cliques will not introduce any new edges. Thus $G_\alpha^+ = G_\beta^+$ if u and v are indistinguishable.

(Only if) We show that the same filled graph is not obtained by swapping u and v when u and v are adjacent and distinguishable. Since u and v are adjacent and distinguishable in α, then we know that $N_{G_\alpha^{i-1}}[u] \neq N_{G_\alpha^{i-1}}[v]$. Thus there exists a vertex $x \in N_{G_\alpha^{i-1}}[u] \setminus N_{G_\alpha^{i-1}}[v]$ or there exists a vertex $y \in N_{G_\alpha^{i-1}}[v] \setminus N_{G_\alpha^{i-1}}[u]$. Observe also that $N_{G_\alpha^{i-1}}[u] = N_{G_\beta^{i-1}}[u]$ and that $N_{G_\alpha^{i-1}}[v] = N_{G_\beta^{i-1}}[v]$, since the exact same set of vertices has been eliminated before u and v in both α and β, and thus $G_\alpha^{i-1} = G_\beta^{i-1}$. If vertex x exists, then it follows from Lemma 4 of [17] that edge $xv \in G_\alpha^+$ and $xv \notin G_\beta^+$. If vertex y exists, then it also follows from Lemma 4 of [17] that edge $yu \in G_\beta^+$ and $yu \notin G_\alpha^+$. Thus $G_\alpha^+ \neq G_\beta^+$ if u and v are adjacent and distinguishable. □

Observe that if u and v are indistinguishable in α they are also indistinguishable in β. Note that similar results have been shown for the degrees of such consecutive vertices u and v [8]. Based on the above result, we give an interesting characterization of equivalent orderings in Theorem 1, which can be used as an algorithm to modify a given ordering and generate any desired equivalent ordering. For the proof of Theorem 1 we first need the following lemma.

Lemma 3 *Let $k < n$ be a fixed integer and let α and β be two equivalent orderings on G such that $\alpha^{-1}(i) = \beta^{-1}(i)$ for $1 \leq i \leq k$. Let v be the vertex that satisfies $\beta(v) = k+1$ and $\alpha(v) = k+d$, $d > 1$, and let u be the predecessor of v in α so that $\alpha(u) = \alpha(v) - 1$. Obtain a new ordering σ from α by only swapping u and v. Then $G_\sigma^+ = G_\beta^+$.*

Proof Note first that since α and β coincide in the first k vertices, and v is the vertex numbered $k+1$ in β, the position of v must be larger than $k+1$ in α, as stated in the premise of the lemma. Note also for the rest of the proof that $\alpha(u) = k + d - 1 > k$. We need to show that u and v are either non-adjacent in $G_\alpha^+ = G_\beta^+$ or indistinguishable in α, as the result then follows immediately from Lemma 2. Assume on the contrary that $uv \in E(G_\alpha^+) = E(G_\beta^+)$ and that u and v are distinguishable in α. Thus there exists at least a vertex $x \in N_{G_\alpha^{k+d-1}}(u) \setminus N_{G_\alpha^{k+d-1}}[v]$, or a vertex $y \in N_{G_\alpha^{k+d-1}}(v) \setminus N_{G_\alpha^{k+d-1}}[u]$. If vertex x exists, then $x \notin N_{G_\alpha^{k+d-1}}(v)$, and thus $x \notin N_{G_\alpha^k}(v) = N_{G_\beta^k}(v)$. It follows that $vx \notin E(G_\beta^+)$, since v is eliminated at step $k+1$ and before u and x in β. This contradicts our assumption that $G_\alpha^+ = G_\beta^+$, because $vx \in G_\alpha^+$ since $ux, uv \in E(G_\alpha^+)$, and u is eliminated before v and x. If vertex y exists, then $uy \notin E(G_\alpha^+)$, since $y \notin N_{G_\alpha^{k+d-1}}(u)$, and u is eliminated before v and y in α. Vertex v is eliminated before y and u in β, and this gives the same

contradiction as above, since uv and vy are contained in $E(G^+_\beta)$, and thus $uy \in E(G^+_\beta)$. □

Regarding orderings β and σ in Lemma 3, note in particular that $\sigma^{-1}(i) = \beta^{-1}(i)$ for $1 \leq i \leq k$ and $\sigma(v) = k + d - 1$. Thus β and σ coincide in the first k positions, and v has moved one position closer to its position in β. We are now ready to present our new characterization of equivalent orderings.

Theorem 1 *Two orderings α and β on G are equivalent, i.e., $G^+_\alpha = G^+_\beta$, if and only if β can be obtained by starting from α and repeatedly swapping pairs of consecutive non-adjacent or indistinguishable vertices.*

Proof The if part of the theorem follows directly from Lemma 2. We prove the only if part by showing that it is always possible to make two equivalent orderings identical by repeatedly swapping pairs of consecutive non-adjacent or indistinguishable vertices in one of them. For this we use the following inductive approach.

Induction hypothesis: Let α and β be two equivalent orderings on G. Modify α so that the k first vertices of α coincide with the k first vertices of β, for a $k \in [0, n]$, and the remaining $n - k$ vertices have the same local relative order as before the change. Then $G^+_\alpha = G^+_\beta$.

Base case: When $k = 0$ α is unchanged and thus $G^+_\alpha = G^+_\beta$.

Induction step: Given two equivalent orderings α and β on G such that the k first vertices in α coincide with the k first vertices in β. We have to show that, in α, one of the vertices in positions $k + 1, k + 2, ..., n$ can be moved to position $k + 1$ so that the vertex in position $k + 1$ is the same in α and in β. We will show that this can be achieved by repeatedly swapping pairs of consecutive non-adjacent or indistinguishable vertices in positions $k + 1, k + 2, ..., n$ of α. We use induction also for this purpose.

Induction hypothesis: Given two equivalent orderings α and β on G such that the k first vertices in α coincide with the k first vertices in β. Let v be vertex number $k + 1$ in β, and let $\alpha(v) = k + d$, for a $d \in [1, n - k]$. Move v so that $\alpha(v) = k + d - i$, for an $i \in [0, d - 1]$. Then $G^+_\alpha = G^+_\beta$.

Base case: Let $i = 0$, then α is unchanged and thus $G^+_\alpha = G^+_\beta$.

Induction step: Given two equivalent orderings α and β on G such that the k first vertices in α coincide with the k first vertices in β, and vertex v which is numbered $k + 1$ in β has number $k + d - i$ in α. It follows from Lemma 3 that v can be swapped with its predecessor in α and thus moved to position $k + d - (i + 1)$ in α and G^+_α remains unchanged.

We have thus proved that, the first vertex v of β which does not coincide with the vertex in the same position of α, can be repeatedly swapped with its predecessor in α until it reaches the same position in α as in β without changing the filled graph, and that this can again be repeated for all vertices. By Lemma 2 all swapped vertices are non-adjacent and indistinguishable, which completes the proof. □

The following corollary is a direct consequence of Theorem 1.

Corollary 1 *Two orderings α and β on G are equivalent if and only if there is a sequence of orderings $\sigma_1, \sigma_2, ..., \sigma_k$ where $\sigma_1 = \alpha$, $\sigma_k = \beta$, and σ_{i+1} is obtained from σ_i by swapping two vertices, such that $G^+_{\sigma_i} = G^+_{\sigma_{i+1}}$, for $1 \leq i < k$.*

We would also like to draw the reader's attention to the following result which is implicit in [14], and which is now a direct corollary of Theorem 1.

Corollary 2 *Given a chordal graph G and a perfect elimination ordering α of G, an ordering β is a perfect elimination ordering of G if and only if β can be obtained by starting from α and repeatedly swapping pairs of consecutive non-adjacent or indistinguishable vertices.*

4 k-Optimal elimination orderings

In this section, we put the results of the previous section in a practical context with the purpose of modifying a given ordering α to reduce the resulting fill. As we have seen, swapping two consecutive non-adjacent or indistinguishable vertices of α does not change the fill. Consequently, if we want to swap consecutive vertices to reduce fill, these should be distinguishable and neighbors in G^+_α. The reason why we only consider vertices that appear consecutively in an elimination ordering is that it is then easy to do a local computation to find the change in the number of fill edges, without having to recompute the whole filled graph. This saves time for practical implementations. The following lemma gives the change in the number of fill edges when two adjacent vertices are swapped.

Lemma 4 *Given G and α, let uv be an edge of G^+_α such that $\alpha(u) = \alpha(v) - 1 = i$. Let β be the ordering that is obtained from α by swapping u and v. Then $|E(G^+_\alpha)| - |E(G^+_\beta)| = |N_{G^{i-1}_\alpha}(u)| - |N_{G^{i-1}_\alpha}(v)|$.*

Proof Consider the elimination graph G^{i-1}_α. Since uv is an edge of G^+_α, it is also an edge of G^{i-1}_α. If we at step i eliminate u (which corresponds to ordering α), the set of fill edges added incident to v by u is $N_{G^{i-1}_\alpha}[u] \backslash N_{G^{i-1}_\alpha}[v]$. If on the other hand, v is eliminated at step i (which corresponds to ordering β), the set of fill edges added adjacent to u by v is $N_{G^{i-1}_\alpha}[v] \setminus N_{G^{i-1}_\alpha}[u]$. It follows from Lemma 4 of [17] that no other fill is changed since the set of vertices eliminated previous to every vertex $z \in V \setminus \{u, v\}$ are the same for α and β. Thus the difference is $|N_{G^{i-1}_\alpha}[u]| - |N_{G^{i-1}_\alpha}[v]| = |N_{G^{i-1}_\alpha}(u)| - |N_{G^{i-1}_\alpha}(v)|$. □

Although uv is an edge of G^+_α, it might be that u and v do not appear consecutively in α but they appear consecutively in an equivalent ordering β. Since both α and β result in the same filled graph G^+_α, we want to be able to check all edges uv that appear consecutively in α or in an equivalent ordering β. Indeed, we will generalize this idea from 2 to k. Instead of only considering pairs of adjacent vertices, we will consider cliques K of size k whose vertices appear consecutively in the given ordering α or in an equivalent ordering β,

and examine whether a local permutation of the vertices of K can reduce the resulting fill. Note that the vertices of any k-clique in a chordal graph can be ordered with numbers $n-k+1, n-k+2, ..., n$ in a peo [18], and thus we can always find an equivalent ordering β in which the vertices of a given k-clique are ordered consecutively. As a consequence of the results of the previous section, β can be computed from α by swapping pairs of consecutive non-adjacent or distinguishable vertices. However, since K is a clique in G_α^+, we will show that we are able to compute β more efficiently using a clique tree of G_α^+, rather than this more general characterization. Before that, we summarize the above ideas with a definition and an algorithm.

We will always start with making the given ordering α minimal by removing unnecessary edges from G_α^+ by using an appropriate algorithm for this purpose. When α is minimal, we know that no ordering can result in a filled graph that is a proper subgraph of G_α^+. Thus reducing the number of fill edges further can only be achieved if some fill edges are removed and some new fill edges are introduced.

Definition 1 An elimination ordering α is *k-optimal* on G if obtaining an equivalent ordering β where the vertices of K appear consecutively, and then locally permuting the vertices of K, cannot result in less fill for any k-clique K in G_α^+.

Algorithm Minimal k-optimal ordering
Input: A graph G and an ordering α.
Output: A minimal elimination ordering β of G where $|E(G_\beta^+)| \leq |E(G_\alpha^+)|$.
$\beta = \alpha$;
repeat
  Compute a minimal ordering σ such that $E(G_\sigma^+) \subseteq E(G_\beta^+)$;
  $\beta =$ Compute-k-optimal(G, σ);
until β is minimal and k-optimal

The call Compute-k-optimal(G, σ) returns a k-optimal ordering based on σ. In the next section, we describe how to implement this function. Note that computing a minimal ordering removes the unnecessary fill edges from the filled graph, and never adds new fill edges. Computing a k-optimal ordering, however, will both remove and add fill edges, and thus the resulting ordering after this step is not necessarily minimal. By the definition of k-optimality, this algorithm always produces an ordering β with less fill than that of α unless α is already minimal and k-optimal.

5 Implementation details and experimental results

We now describe a practical implementation of a variant of the above algorithm to exhibit the potential of the proposed ideas. Given G and σ, in order to implement the function Compute-k-optimal(G, σ) correctly, we need to be able to do the following subtasks for each k-clique K of G_σ^+: 1. Compute

every equivalent ordering where the vertices of K are ordered consecutively. 2. Try all possible local permutations of the vertices of K in each of these equivalent orderings. Recall however that we need not consider k-cliques in which all vertices are pairwise indistinguishable, and we need not consider local permutations that result in equivalent orderings. Still, the remaining work is too time consuming for a practical algorithm, and in the practical version we generate several but *not* all equivalent orderings in which the vertices of K are ordered consecutively. Furthermore, we only test a single local permutation of K; one that we suspect might give the highest fill reduction. For both of these subtasks, we use clique trees. The vertices of every k-clique K must appear together in at least one tree node of any clique tree of G_σ^+, and subsets of K that appear in other tree nodes will give us various local permutations that we will try. Then using the clique tree, it is easy to compute orderings where the vertices of K appear together and follow the chosen permutations.

Observe that every leaf in any clique tree of G_α^+ contains a simplicial vertex, which can be eliminated as the first vertex in an equivalent ordering. Let us describe how we construct equivalent orderings for a clique K. Let T be a clique tree of G_α^+. A subtree or a forest of subtrees remains when we remove every tree node in T containing K. One equivalent ordering is now created for each remaining subtree. Let T' be one such subtree, and let T'' be the rest of T when this subtree is removed. Let V' be the set of vertices of G contained in the tree nodes of T' and not contained in the tree nodes of T''. An equivalent ordering is now obtained by eliminating the vertices of $V \setminus (V' \cup K)$, then the vertices of K, and finally the vertices of V'. Within each of these three sets we eliminate the vertices in a simplicial order.

There exists at least one vertex in K which is not adjacent to any vertex in V'. If this was not the case, then K would be contained in the tree node of T' which is adjacent in T to a tree node containing K, which is a contradiction since none of the tree nodes of T' contains K. Eliminating such a vertex will make the remaining vertices of K into a clique. We want to avoid this for two reasons. The first is that we want to obtain a new graph, and the second is that we do not want to destroy the possibility of removing edges in K. A simple solution to this problem is to eliminate these vertices last when K is eliminated. So the new order is created as follows: Eliminate the vertices of $V \setminus (V' \cup K)$ in a simplicial order, then eliminate the vertices of K that are adjacent to a vertex in V' in a random order, then eliminate the rest of the vertices of K in any order. Finally eliminate the vertices of V' in a simplicial order. This will give us an ordering that introduces new fill edges, and possibly remove some.

In order to achieve good orderings, it is important to start the process with an ordering that already produces low fill. We have tested our implementation with Minimum Degree as the starting ordering. Minimum Degree orderings are in general neither minimal nor k-optimal for any $k > 1$, although they tend to be often minimal in practice [1,16]. However, even Minimum Degree orderings that are minimal need not be 2-optimal, as shown in Fig.1. Thus Minimum Degree is a good starting ordering for our purposes. Table 1 and Fig. 2 show the reductions in fill achieved by our implementation when trying

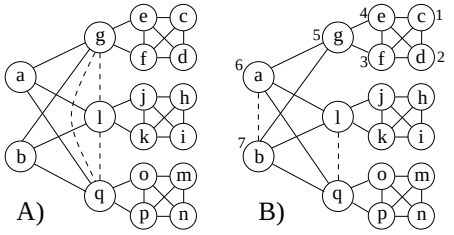

Fig. 1 The solid edges are the original edges of the given graph, while the dashed edges are fill edges resulting from the elimination process. In figure A we use the elimination ordering given by the alphabetic ordering of the vertices, which is minimal and also a Minimum Degree ordering. Consider the equivalent ordering $\{c, d, e, f, g, a, b, h, i, ..., q\}$ and the clique $\{a, g\}$. Swapping a and g gives us a new ordering $\{c, d, e, f, a, g, b, h, i, ..., q\}$ which contains one less fill edge, as shown in figure B. Thus the Minimum Degree ordering of A is minimal but not 2-optimal.

to make an initial Minimum Degree ordering k-optimal for $k \in \{2, 3, 4\}$ on a collection of sparse matrices downloaded from Matrix Market [13]. Columns 6, 8, and 10 of the table show the achieved reduction in the number of fill edges, whereas columns 7, 9, and 11 show the reduction in percentage. Fig. 2 shows the reductions in percentage only. As can be seen, the reduction is substantial in several cases. For four other graphs that we tested, bcsstk04, lund_a, lund_b, and nos1, no reduction was achieved for these k-values, and to save space, we omit these graphs in the table and the following figure.

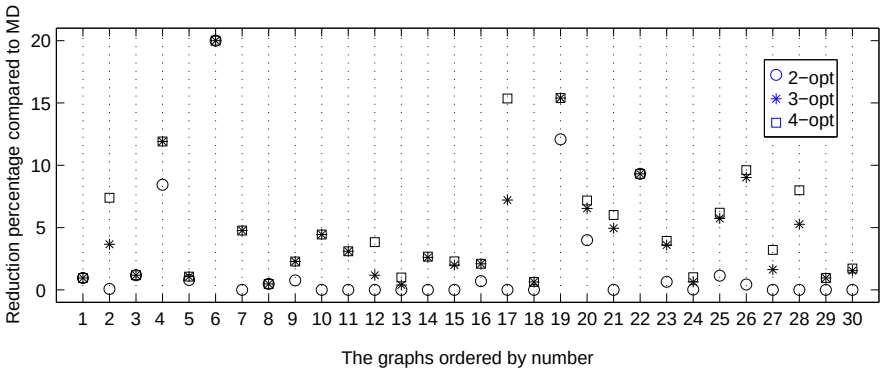

Fig. 2 This figure only shows the reduction in percentage.

6 Concluding remarks

The main theoretical contribution of this paper has been a new characterization of equivalent reorderings that makes it possible to generate such orderings through small local changes from a given ordering. We have also

No	Name	$\|V\|$	$\|E\|$	MD fill	2-opt	2-opt %	3-opt	3-opt %	4-opt	4-opt %
1	494_bus	494	586	314	3	0.96	3	0.96	3	0.96
2	ash292	292	958	1392	1	0.07	51	3.66	103	7.40
3	bcspwr03	118	179	85	1	1.18	1	1.18	1	1.18
4	bcspwr04	274	669	403	34	8.44	48	11.91	48	11.91
5	bcspwr05	443	590	375	3	0.80	4	1.07	4	1.07
6	bcsstk03	112	265	10	2	20.00	2	20.00	2	20.00
7	bcsstk05	153	1135	1197	0	0	57	4.76	57	4.76
8	bcsstk20	485	1328	412	2	0.49	2	0.49	2	0.49
9	bcsstk22	138	282	263	2	0.76	6	2.28	6	2.28
10	can_144	144	576	225	0	0	10	4.44	10	4.44
11	can_161	161	608	1551	0	0	48	3.09	48	3.09
12	can_187	187	652	1277	0	0	15	1.17	49	3.84
13	can_229	229	774	2008	0	0	8	0.40	20	1.00
14	can_256	256	1330	1913	0	0	50	2.61	51	2.67
15	can_268	268	1407	2096	0	0	42	2.00	48	2.29
16	can_292	292	1124	1143	8	0.70	24	2.10	24	2.10
17	dwt_162	162	510	638	0	0	46	7.21	98	15.36
18	dwt_193	193	1650	2367	0	0	15	0.63	15	0.63
19	dwt_198	198	602	513	62	12.09	79	15.40	79	15.40
20	dwt_209	209	767	1102	44	4.00	72	6.53	79	7.17
21	dwt_221	221	704	849	0	0	42	4.95	51	6.01
22	dwt_234	234	306	204	19	9.31	19	9.31	19	9.31
23	dwt_245	245	608	610	4	0.66	22	3.61	24	3.93
24	dwt_307	307	1108	3421	1	0.03	22	0.64	35	1.02
25	dwt_310	310	1069	1920	22	1.15	110	5.73	119	6.20
26	dwt_346	346	1443	3516	15	0.43	317	9.02	338	9.61
27	dwt_361	361	1296	3366	0	0	55	1.63	108	3.21
28	dwt_419	419	1572	3342	0	0	176	5.27	267	8.00
29	lshp_265	265	744	2185	0	0	21	0.96	21	0.96
30	lshp_406	406	1155	4074	0	0	63	1.55	70	1.72

Table 1 We would like to remark that in order to avoid "lucky" reductions, we ran Minimum Degree on 5 random permutations of each graph, and chose the ordering that gave the lowest fill as the starting ordering.

explained how this knowledge can be exploited to reduce fill in sparse Gaussian elimination.

We would like to stress that the goal of our numerical experiments has been to exhibit the potential of the proposed algorithm to find good quality orderings with respect to fill. We have not optimized the the running time of our code, thus we do not find it useful to give the running time of computing a 3-optimal or a 4-optimal ordering compared to the Minimum Degree algorithm whose code has been thoroughly optimized through the last decades. Although computing 3-optimal and 4-optimal orderings are computationally heavy tasks that require more time than Minimum Degree, we believe that practical implementations using approximations are possible. This is an interesting topic for further research.

Simple examples exist which show that a k-optimal ordering is not necessarily $(k-1)$-optimal or $(k+1)$-optimal. However, we suspect that when checking k-optimality, if we also check all maximal cliques in G_α^+ of size less

than k, then we can also guarantee $(k-1)$-optimality. Note that the number of maximal cliques in G_α^+ is $O(n)$, and thus this extra work is substantially less than the work of checking all (both maximal and non-maximal) k-cliques that the definition requires. We leave this as an open question.

References

1. J. R. S. Blair, P. Heggernes, and J. A. Telle. A practical algorithm for making filled graphs minimal. *Theor. Comput. Sci.*, 250:125–141, 2001.
2. J. R. S. Blair and B. W. Peyton. An introduction to chordal graphs and clique trees. In *Sparse Matrix Computations: Graph Theory Issues and Algorithms*, pages 1–30. Springer Verlag, 1993.
3. E. Dahlhaus. Minimal elimination ordering inside a given chordal graph. In Proceedings of WG 1997, pages 132–143. Springer Verlag, 1997. LNCS 1335.
4. G. A. Dirac. On rigid circuit graphs. *Anh. Math. Sem. Univ. Hamburg*, 25:71–76, 1961.
5. D. R. Fulkerson and O. A. Gross. Incidence matrices and interval graphs. *Pacific Journal of Math.*, 15:835–855, 1965.
6. F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *J. Combin. Theory Ser. B*, 16:47–56, 1974.
7. J. A. George. Nested dissection of a regular finite element mesh. *SIAM J. Numer. Anal.*, 10:345–363, 1973.
8. J. A. George and J. W. H. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1981.
9. J. A. George and J. W. H. Liu. The evolution of the minimum degree ordering algorithm. *SIAM Review*, 31:1–19, 1989.
10. P. Heggernes and Y. Villanger. Efficient implementation of a minimal triangulation algorithm. In *Algorithms - ESA 2002*, pages 550–561. Springer Verlag, 2002. LNCS 2461.
11. L. Ibarra. Fully dynamic algorithms for chordal graphs. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1999.
12. J. W. H. Liu. Equivalent sparse matrix reorderings by elimination tree rotations. *SIAM J. Sci. Stat. Comput.*, 9(3):424–444, 1988.
13. R. Boisvert, R. Pozo, K. Remington, B. Miller, and R. Lipman. *NIST Matrix Market*. http://math.nist.gov/MatrixMarket/.
14. L. S. Chandran, L. Ibarra, F. Ruskey, and J. Sawada. Generating and characterizing the perfect elimination orderings of a chordal graph. *Theor. Comput. Sci.*, 307:303–317, 2003.
15. S. Parter. The use of linear graphs in Gauss elimination. *SIAM Review*, 3:119–130, 1961.
16. B. W. Peyton. Minimal orderings revisited. *SIAM J. Matrix Anal. Appl.*, 23(1):271–294, 2001.
17. D. Rose, R. E. Tarjan, and G. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM J. Comput.*, 5:146–160, 1976.
18. D. J. Rose. Triangulated graphs and the elimination process. *J. Math. Anal. Appl.*, 32:597–609, 1970.
19. Y. Villanger. Lex M versus MCS-M. *Discrete Math.*, to appear.
20. M. Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM J. Alg. Disc. Meth.*, 2:77–79, 1981.