Angewandte Mathematik und
Informatik
Universität zu Köln

Report No. 353

An Improved Linear Time Algorithm for Minimal Elimination Ordering in Planar Graphs that is Parallelizable

by
Elias Dahlhaus

1999

Partially supported by ESPRIT Long Term Research Project Nr. 20244 (ALCOM-IT).

Institut für Informatik, Universität zu Köln,
Pohligstrasse 1, 50969 Köln, Germany

Abstract

We present an alternative linear time algorithm that computes an ordering that produces a fill-in that is minimal with respect to the subset relation. It is simpler than the algorithm in [6] and is easily parallelizable. The algorithm does not rely on the computation of a breadth-first search tree.

1 Introduction

One of the major problems in computational linear algebra is that of sparse Gauss elimination. The problem is to find a pivoting, such that the number of zero entries of the original matrix that become non zero entries in the elimination process is minimized. In case of symmetric matrices, we would like to restrict pivoting along the diagonal. The problem translates to the following graph theory problem [16].

Minimum Elimination Ordering

For an ordering $<$ on the vertices, we consider the fill-in graph $G'_< = (V, E')$ of $G = (V, E)$. $G'_<$ contains first the edges in E and secondly two vertices x and y form an edge in $G'_<$ if they have a common smaller neighbor in $G'_<$. *The problem of Minimum Elimination ordering is, given a graph $G = (V, E)$, find an ordering $<$, such that $G'_<$ has a minimum number of fill-in edges.* Note that this problem is NP-complete [22].

For this reason, one considers also the following relaxation of the problem.

Minimal Elimination Ordering

Given a graph G, find an ordering $<$, such that the edge set of $G'_<$ is minimal with respect to inclusion. This problem can be solved in $O(nm)$ time [17].

The problem to get an elimination ordering with a small fill-in is in particular interesting for planar graphs, e.g. finite elements (see for example [9]).

In case that $G = G'_<$ ($<$ has no fill-in edges) $<$ is a perfect elimination ordering, and graphs having a perfect elimination ordering are exactly the *chordal graph*, i.e. graphs with the property that every cycle of length greater three has an edge that joins two non consecutive vertices of the cycle.

The Minimal Elimination Ordering problem can be solved in $O(nm)$ time in general [17] and for planar graphs in linear time [6]. For graphs in general, the problem to get a minimal elimination ordering can be parallelized. The time bound is $O(\log^3 n)$ on a CRCW-PRAM, and the processor bound is $O(nm)$ [8]. The problem of the algorithm in [6] is that it strongly relies on breadth-first search. A parallelization can therefore only be done through the quite complicated breadth-first search algorithm of Klein for planar planar graphs [13].

In this paper, a quite simple linear time algorithm for the minimal elimination ordering problem for planar graphs is presented. The algorithm is also easily parallelizable.

In section 2, we introduce the notation. In section 3, we determine a first "approximation" of a minimal elimination ordering using a spanning tree. In section 4, we add edges to the given planar graph that are fill-in edges of any extension of the minimal elimination approximation computed in 3. We call these edges *level diagonals*. Adding level diagonals does not destroy planarity. In section 5 we discuss the structure of one level of the minimal elimination ordering approximation computed in section 3. In section 6 we transform the discussions in section 5 into an algorithm to refine each minimal elimination ordering approximation level. The final section gives some concluding remarks.

2 Notation

A *graph* $G = (V, E)$ consists of a *vertex set* V and an *edge set* E. Multiple edges and loops are not allowed. The edge joining x and y is denoted by xy.

We say that x is a *neighbor* of y iff $xy \in E$. The set of neighbors of x is denoted by $N(x)$ and is called the *neighborhood*. More general, for a vertex set V', $N(V') = \{y \in V \setminus V' | \exists x \in V' : xy \in E\}$.

For a subset V' of V, $G[V']$ is the subgraph of G induced by V' and $G - V'$ is the subgraph of G induced by $V \setminus V'$.

Trees are always directed to the root. The notion of the *parent*, *child*, *ancestor*, and *descendent* are defined as usual.

We denote by n the number of vertices and by m the number of edges of G.

A graph is called *chordal* iff each cycle of length greater than three has a chord, i.e. an edge that joins two nonconsecutive vertices of the cycle. Note that chordal graphs are exactly those graphs having a *perfect elimination ordering* $<$, i.e. for each vertex v the neighbors $w > v$ induce a complete subgraph, i.e. they are pairwise joined by an edge [10].

Note that in any chordal graph, the number of maximal cliques is bounded by n and the number of pairs (x, c) such that x is in the clique c is bounded by the number of edges and vertices.

The problem of minimum (minimal) elimination ordering is therefore equivalent to the problem to find, given a graph $G = (V, E)$, an extension E' of E, such that (V, E') is chordal and $|E'|$ is minimum (E' is minimal with respect to the subset relation).

A *cut* c of $G = (V, E)$ is a subset of V, such that $G - c$ has at least two connected components and the neighborhood of at least two connected components of $G - c$ is c.

3 Minimal Elimination Ordering Approximation through a Spanning Tree

We start with a result that is also given in [7]. We call an ordered partition $V_1, \ldots, V_k$ *an approximation* of an ordering $<$ if with $x \in V_i$, $y \in V_j$, and $i < j$, we have $x < y$ and an ordering $<$ *compatible* with $V_1, \ldots, V_k$ if $V_1, \ldots, V_k$ is an approximation of $<$.

Theorem 1 *Let T be a spanning tree of G and $v_1, \ldots, v_n$ be an enumeration of the vertex set of T, such that each final segment $\{v_i, \ldots, v_n\}$ induces a subtree of T (e.g. postorder enumeration). Let*

$$V_i = \{y | yv_i \in E \textit{ and for } j > i\, yv_j \notin E\}$$

Then $V_1, \ldots, V_n, \{v_n\}$ is an approximation of a minimal elimination ordering.

Sketch of Proof: By [15], a maximal set $\mathcal{C}$ of non crossing cuts of G is the set of cuts of a minimal extension of G to a chordal graph, i.e. there is a minimal elimination ordering $<$, such that the set of cuts of the chordal graph $G'_<$ is exactly $\mathcal{C}$. It is easily checked that for any connected component C of $\bigcup_{j<i} V_j$, the set of neighbors of C outside C is a cut. Q.E.D.

Note that the ordered partition $(V_1, \ldots, V_n, \{v_n\})$ can be determined in $O(n+m)$ time and by a CRCW-PRAM in logarithmic time with a linear processor number [18].

Now we consider only the subsequence of nonempty partion elements and denote it by $V_1, \ldots, V_k$, i.e. $V_k = \{v_n\}$. Nevertheless, V_i is the set of neighbors of a vertex $v_i \in V_{i+1} \cup \ldots \cup V_k$ that are not in $V_{i+1} \cup \ldots \cup V_k$. $v_1, \ldots, v_{k-1}$ is also called the *generating sequence* of $V_1, \ldots, V_k$.

For graphs in general, the further refinement has to be done by the general algorithm of [17], i.e. we get no better theoretical improvement than a time bound of $O(nm)$, to get a minimal elimination ordering.

For planar graphs, we get a more efficient solution.

4 Introducing Level Diagonals

As in [6], we would like that the neighborhoods of connected components of $V_{<i} = \bigcup_{j<i} V_j$ form cycles. Let $l(v)$ be the i, such that $v \in V_i$.

Initially $E' = E$ is the edge set of G. For a face f, let $(v_0, \ldots, v_{l-1})$ be the clockwise cyclic enumeration of the vertices of f. We add an edge v_iv_j to E' if for all i' with $i' = i + 1 \, mod\, l, i + 2\, mod\, l, \ldots, j - 1\, mod\, l, j$, $l(v_{i'}) < l(v_i)$ and $l(v_{i'}) < l(v_j)$. In other words v_iv_j is put into E' if one of the two paths from v_i to v_j of the cycle surrounding the face f contains only inner vertices that are in V_ν with $\nu < l(v_i)$ and $\nu < l_(v_j)$.

$G' = (V, E')$ is called the *level diagonal graph* of $V_1, \ldots, V_k$. *It is easily checked that G' is planar.*

Moreover, *we can show that all edges of G' that are not in G are fill-in edges of any ordering that is compatible with $V_1, \ldots, V_k$.* Observe that *the neighborhood of any connected component of $G[V_1 \cup \ldots \cup V_i]$ in $V_{i+1} \cup \ldots \cup V_k$ is a cycle of G'.*

Lemma 1 *The edges of E' can be determined in linear time and in $O(\log n)$ time with $O(n/\log n)$ processors on a CRCW-PRAM.*

Proof: Consider any face f. We select a vertex u_0 such that $l(u_0)$ is maximum. For each vertex u_i of f, we determine the next vertex $s(u_i)$ before u_i and the next vertex $g(u_i)$ after u_i in a level V_j with $j \geq l(u_i)$. The indices $s(i)$ and $g(i)$ can be determined in linear time sequentially and in logarithmic time with a linear workload on a CRCW-PRAM [21].

Q.E.D.

5 The Structure of the Levels V_i

Let $v_1, \ldots, v_k$ be the generating sequence of $V_1, \ldots, V_k$.

Note that also in G', V_i is the set of neighbors of v_i in G' that are not neighbors of any v_j with $j > i$.

One can also observe the following.

Lemma 2 *If v and w are in V_i and in the same connected component of $G'[\bigcup_{j \leq i} V_j]$ then they are in the same connected component of $G'[V_i]$.*

Let F be a connected component of $G'[V_i]$.

We call a vertex of $G' - F$ an *inner vertex* of the face f of $G'[F]$ if it appears inside f in the given planar embedding. The vertices *of f* are the vertices at the boundary of f.

Observe that there is only one face of $G'[F]$ that has vertices of (the connected set) $\bigcup_{j>i} V_j$ as inner vertices. This face is called the *outer face* of F. Moreover, all vertices of F are vertices of the outer face of F, i.e. $G'[F]$ is an *outer-planar* graph.

Observe also that there if only one face f_F of $G[F \cup \{v_i\}]$ that contains vertices of $\bigcup_{j>i} V_j \setminus \{v_i\}$ as inner vertices. The vertices of F of this face form a subpath of the cycle surrounding f_F and are called the *outer vertices of F* (see figure 1).

Observe that only outer vertices of F have neighbors in $\bigcup_{j>i} V_j$ that are different from v_i.

Outer vertices having neighbors in $\bigcup_{j>i} V_j$ are called *strong outer vertices.* Let $w_1, \ldots, w_m$ be the path of outer vertices of F and w_ν and w_μ be consecutive strong outer vertices of F. Then $w_\nu \ldots w_\mu$ is a subpath of a face f_{w_ν, w_μ} of

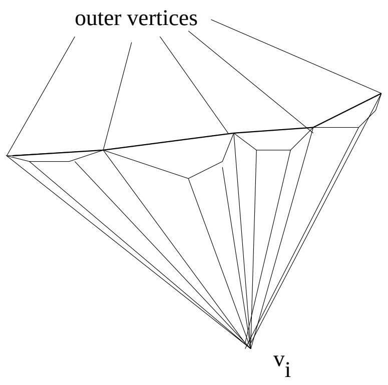

Figure 1: Outer Vertices of a Connected Component F of V_i

$G'[\bigcup_{j\geq i} V_j]$. Such a face is called a *border face* of F. The subsequence of strong outer vertices is denoted by $p_1, \ldots, p_l$. Note that the border faces are uniquely determined by their strong outer vertices p_ν and $p_{\nu+1}$.

The *essential faces* of F are the non-outer and the border faces of F.

One can observe immediately (compare also [6]) the following.

Lemma 3 *If an essential face f of F contains inner vertices in $\bigcup_{j<i} V_j$ then the vertex set of f is made complete in the fill-in of any ordering that is compatible with $V_1, \ldots, V_k$.*

If an essential face f of F has no inner vertices of $\bigcup_{j<i} V_j$, we call f *empty*. Otherwise we call f *full*.

It can be checked in logarithmic time and with a linear workload whether a face is full. We only have to select a vertex u of f and to check whether there are neighbors of u in G' that are between the two neighbors of u in f in the clockwise enumeration of the neighborhood of u.

We extend $G'[F]$ to a graph G^*_F containing additional edges $p_j p_{j+1}$ if the border face containing the consecutive strong outer vertices p_j and p_{j+1} is full. Observe that G^*_F is still planar and that all vertices of G^*_F are on the outer face of G^*_F, i.e. also G^* is outer-planar. Note that the faces of G^*_F are the faces of $G'[F]$ and the faces containing all outer vertices of F that are between two consecutive strong outer vertices p_j and p_{j+1} that share a full face. These additional faces of G^*_F are also called full faces of G^*_F.

The graph G''_F is the following extension of G^*_F: full faces of G^*_F are made complete and empty (non-outer) faces of G^*_F are triangulated. To get a linear space representation of G''_F, we determine an extension G^{**}_F where empty faces are triangulated but full faces are not made complete.

As in [6], we can show the following.

Lemma 4 *G''_F is a chordal graph.*

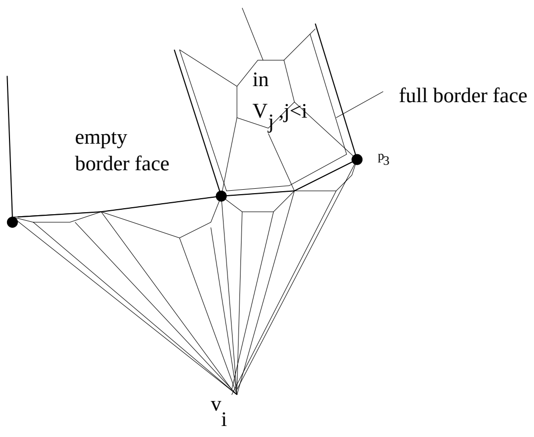

Figure 2: Strong Outer Vertices and Full and Empty Border Faces

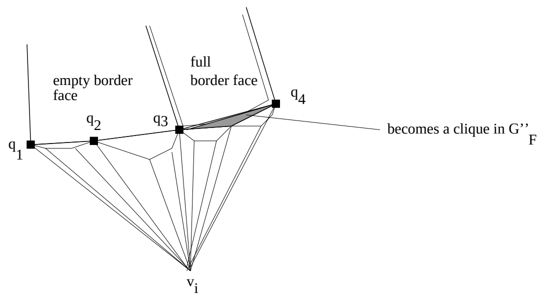

Figure 3: Proper Outer Vertices

The strong outer vertices and full and empty border faces can be observed in figure 2.

We call an outer vertex q of F a *proper* outer vertex of F if it is a strong outer vertex or on the path between two strong consecutive strong outer vertices that that do not share a full border face, i.e. if we add v_i and all edges v_iw with $w \in F$ to G_F^{**} then q remains on the outer face.

Proper outer vertices and G''_F are illustrated in figure 3.

6 Algorithm for Further Refinement

To determine a minimal elimination ordering, we determine for each F, a perfect elimination ordering of G''_F that extends to a minimal elimination ordering of the graph H_F consisting of the vertices and edges of G''_F, the neighborhood of F in G' as a complete set, and all edges vw between F and its neighborhood that are in G' or that share a full border face of F.

The subsequence of the outer vertices $w_1, \dots w_m$ containing exactly the proper outer vertices is denoted by $q_1, \dots, q_k$.

We select an appropriate $q_j = w_\nu$ as maximum. We determine a perfect elimination ordering $<_F$ of G''_F, such that

1. non-outer vertices are smaller than outer vertices of F,
2. non-proper outer vertices are smaller than proper outer vertices, and
3. q_j is the maximum.

Such an ordering is called *consistent* with q_j.

Lemma 5 *Given q_j and G^{**}_F, a consistent ordering can be determined in logarithmic time and linear workload.*

Sketch of Proof: First we determine an ordering $<_F$ of the proper outer vertices, such that for $\nu < j$, $q_\nu <_F q_{\nu+1}$ and for $\nu > j$, $q_\nu <_F q_{\nu-1}$. Then we order the non-proper outer vertices in any how.

It remains to order the non-outer vertices of F. We determine a clique tree T_F of G''_F, i.e. a face tree of G^{**}_F. The nodes of the clique tree are the faces and the bridges of G^{**}_F. If f_1 and f_2 share an edge then they are joined by an edge of T_F. If $G^{**}_F - \{v\}$ is has more than one connected component then we select for each connected component c a face or bridge f_c of c that contains v. We join the faces and bridges f_c associated with v by edges, such that they form a tree. Note that the faces and bridges containing any vertex v form a subtree of T_F. T_F can be determined in logarithmic time with a linear workload.

We select a face containing q_j as the root of T_F. We determine a postorder $<_T$ on the nodes of T_F. For each non-outer vertex v, let f_v be the $<_T$-maximum face or bridge containing v. We determine an ordering $<_F$ on the non-outer vertices of F, such that with $f_v <_T f_w$, we have $v <_F w$.

Q.E.D.

Lemma 6 *1. If $\mu < j$ then $q_\mu <_F q_{\mu+1}$, and if $\mu > j$ then $q_\mu <_F q_{\mu-1}$.*

2. *In H_F, all fill-in edges of $<_F$ are between F and $H_F \setminus F$, only proper outer vertices are incident with fill-in edges of $<_F$.*
3. *In the fill-in of $<_F$, the neighbors of q_j in $H_F - F$ are the neighbors of any q_ν in $H_F - F$.*
4. *For $\nu < j$ ($\nu > j$), the fill-in neighborhood of q_ν in $H_F - F$ consists of the neighborhood of any q_μ, such that $\mu \leq \nu$ ($\mu \geq \nu$).*

We denote by $N_{\leq}(q_\nu)$ the neighborhood of all q_μ with $\mu \leq \nu$ in $H_F - F$, and by $N_{\geq}(q_{nu})$ the neighborhood all q_μ with $\mu \geq \nu$ in $H_F - F$. That means that the fill-in neighborhood of any q_ν with $\nu < j$ is $N_{\leq}(q_\nu)$ and the fill-in neighborhood of any q_ν with $\nu > j$ is $N_{\geq}(q_\nu)$.

We have to find a q_j, such that the corresponding ordering $<_F$ is a minimal elimination ordering of H_F.

We call a q_j *inappropriate* if $N_{\leq}(q_j) \subset N_{\geq}(q_{j+1})$ or $N_{\geq}(q_j) \subset N_{\leq}(q_{j-1})$, for some $\nu < j$. Otherwise, q_j is called *appropriate.*

Lemma 7 *If q_j is appropriate then the corresponding ordering $<_F$ that is consistent with q_j is a minimal elimination ordering of H_F.*

We try to find an appropriate p_j (i.e. a strong outer vertex that is appropriate).

We determine the neighbor $left(F)$ of p_1 that is the next neighbor of p_1 in some V_l, $l > i$ after v_i in the clockwise enumeration of the neighborhood of p_1 and the last neighbor $right(F)$ in some V_l, $l > i$ before v_i in the clockwise enumeration of the neighborhood of the last element p_k of the sequence of the strong outer vertices.

If F has only one neighbor $\neq v_i$ in some V_l, $l > i$ then any strong outer vertex is appropriate.

We assume that this is not the case. Then we can show the following.

Lemma 8 *p_j is inappropriate if and only if one of the following conditions is satisfied.*

1. *p_{j+1} has $left(F)$ as a neighbor.*
2. *The border face of p_j and p_{j+1} is full and contains $left(F)$.*
3. *p_{j-1} has $right(F)$ as a neighbor.*
4. *The border face of p_j and p_{j-1} is full and contains $right(F)$.*

Note that this lemma relies strongly on the planarity of G'.

It is also easily checked that we can determine appropriateness in constant time with a linear workload.

We finally get the following result.

Theorem 2 *For planar graphs, a minimal elimination ordering can be determined in linear time. Moreover it can be done in logarithmic time with a linear processor number on a CRCW-PRAM.*

7 Conclusions

This algorithm is an improvement of the algorithm in [6] for the following reasons. As also mentioned in the introduction, the algorithm is easily parallelizable. Moreover, the algorithm might be embeddable into a nested dissection procedure [2]. It might be possible to get good results if one makes a recursive partition of the vertex set as done in many nested dissection procedures. The recursive partition defines a metric in an obvious way. One might build up a minimum spanning tree, and one gets a first approximation of a minimal elimination ordering. This approach might also be helpful to solve the sandwich problem to get a minimal elimination inside a given chordal supergraph (see [3, 5]).

References

[1] K. Abrahamson, N. Dadoun, D. Kirkpatrick, T. Przyticka, A Simple Parallel Tree Contraction Algorithm, *Journal of Algorithms* 10 (1988), pp. 287-302.

[2] A. Agrawal, P. Klein, R. Ravi, Cutting Down on Fill-in Using Nested Dissection, in *Sparse Matrix Computations: Graph Theory Issues and Algorithms*, A. George, J. Gilbert, J.W.-H. Liu ed., IMA Volumes in Mathematics and its Applications, Vol. 56, Springer Verlag, 1993, pp. 31-55.

[3] J. Blair, P. Heggernes, J.A. Telle, *Making an Arbitrary Filled Graph Minimal by Removing Fill Edges*, Algorithm Theory-SWAT '96, R. Karlsson, A. Lingas ed., LLNCS 1097, pp. 173-184.

[4] P. Bunemann, *A Characterization of Rigid Circuit Graphs*, Discrete Mathematics 9 (1974), pp. 205-212.

[5] E. Dahlhaus, *Minimal Elimination Ordering inside a Given Chordal Graph*, WG 97 (R. Möhring ed.), LLNCS 1335, pp. 132-143.

[6] E. Dahlhaus, *Minimal Elimination of Planar Graphs*, Algorithm Theory - SWAT'98, LLNCS 1432 (1998), pp. 210-221.

[7] E. Dahlhaus, *Minimal Elimination Ordering for Graphs of Bounded Degree*, submitted.

[8] Elias Dahlhaus, Marek Karpinski, *An Efficient Para llel Algorithm for the Minimal Elimination Ordering (MEO) of an Arbitr ary Graph*, Theoretical Computer Science 134 (1994), pp. 493-528.

[9] M. Eiermann, O. Ernst, W. Queck, *Finite Element Tutorial*, TU-Bergakademie Freiberg.

[10] M. Farber, *Characterizations of Strongly Chordal Graphs*, Discrete Mathematics 43 (1983), pp. 173-189.

[11] F. Gavril, *The Intersection Graphs of Subtrees in Trees Are Exactly the Chordal Graphs*, Journal of Combinatorial Theory Series B, vol. 16(1974), pp. 47-56.

[12] J. Gilbert, R. Tarjan, *The Analysis of a Nested Dissection Algorithm*, Numerische Mathematik 50 (1987), pp. 427-449.

[13] P. Klein, S. Subramanian, *A Linear-Processor Polylog-Time Algorithm for Shortest Paths in Planar Graphs*, 34-th FOCS (1993), pp. 259-270.

[14] R. Lipton, R. Tarjan, *A Separator Theorem for Planar Graphs*, SIAM Journal on Applied Mathematics 36 (1979)¡ pp. 177-189.

[15] Parra, A., Scheffler, P., How to use minimal separators for its chordal triangulation, *Proceedings of the* 20^{th} *International Symposium on Automata, Languages and Programming (ICALP'95)*, Springer-Verlag Lecture Notes in Computer Science **944**, (1995), pp. 123–134.

[16] D. Rose, *Triangulated Graphs and the Elimination Process*, Journal of Mathematical Analysis and Applications 32 (1970), pp. 597-609.

[17] D. Rose, R. Tarjan, G. Lueker, *Algorithmic Aspects on Vertex Elimination on Graphs*, SIAM Journal on Computing 5 (1976), pp. 266-283.

[18] Y. Shiloach, U. Vishkin, *An O(log n) Parallel Connectivity Algorithm*, Journal of Algorithms 3 (1982), S. 57-67.

[19] R. Tarjan, U. Vishkin, *Finding Biconnected Components in Logarithmic Parallel Time*, SIAM-Journal on Computing 14 (1984), pp. 862-874.

[20] R. Tarjan, M. Yannakakis, *Simple Linear Time Algorithms to Test Chordality of Graphs, Test Acyclicity of Hypergraphs, and Selectively Reduce Acyclic Hypergraphs*, SIAM Journal on Computing 13 (1984), pp. 566-579.
Addendum: SIAM Journal on Computing 14 (1985), pp. 254-255.

[21] H. Wagener, Triangulating a Monotone Polygon in Parallel, *Computational Geometry and its Applications*, LNCS 333 (1988), pp. 136-142.

[22] M. Yannakakis, Computing the Minimum Fill-in is NP-complete, *SIAM Journal on Algebraic and Discrete Methods* 2 (1981), pp. 77-79.