RAL-TR-2000-031

Multilevel Algorithms for Wavefront Reduction

by

Y.F. Hu[a] and J.A. Scott[b]

Abstract

Multilevel algorithms are proposed for reordering sparse symmetric matrices to reduce the wavefront and profile. A graph representation of the matrix is used and two graph coarsening methods are investigated. A multilevel algorithm that uses a maximal independent vertex set for coarsening and the Sloan algorithm on the coarsest graph is shown to produce orderings that are of a similar quality to those obtained using the best existing combinatorial algorithm (the hybrid Sloan algorithm). Advantages of the proposed algorithm over the the hybrid Sloan algorithm are that it does not require any spectral information and is significantly faster, requiring on average half the CPU time.

Keywords: Wavefront and profile reduction, multilevel algorithm, row ordering, frontal method.

---

[a]Computational Science and Engineering Department, Daresbury Laboratory, Warrington WA4 4AD, England.

[b]Computational Science and Engineering Department, Atlas Centre, Rutherford Appleton Laboratory, Oxon OX11 0QX, England.

August 31, 2000.

Contents

1 Introduction

We consider multilevel algorithms for ordering sparse symmetric matrices for small wavefront and profile. The resulting ordering may be used to construct a row order for use with the row-by-row frontal method applied to a matrix with a symmetric sparsity pattern (see [22]). Since we are primarily concerned with matrices that are positive definite, we work only with the pattern of the matrix and do not take into account permutations needed for stability. In cases where the matrix is non-definite, or is symmetric only in its sparsity pattern, the actual factorization may be more expensive and require more storage.

Minimizing the profile of a matrix is known to be an NP-complete problem [18]. A number of heuristic algorithms have been proposed, including the Cuthill-McKee [4], Reverse Cuthill-McKee [6, 19], Gibbs-King [8], Gibbs-Poole-Stockmeyer [7], and Sloan [26] algorithms. More recently, spectral orderings based on the Fiedler vector of the Laplacian matrix associated with A have been developed [1, 20, 21]. Kumfert and Pothen [17] propose combining the Sloan algorithm with the spectral ordering. The resulting hybrid Sloan algorithm (hereafter referred to as the *Hybrid algorithm*) has been shown to give much better orderings for large problems than either the spectral method or the Sloan method alone. This has been confirmed by Reid and Scott [22], who implemented the Sloan and Hybrid algorithms within the Harwell Subroutine Library [11] code `MC60`.

One reason for the success of the Hybrid algorithm is that the spectral algorithm takes a global view of the graph of the matrix. This global view is fed into the Sloan algorithm as a priority vector, and the Sloan algorithm then performs local optimizations.

The spectral algorithm has also been used in the area of graph partitioning [9, 25]. More recently, researchers have found that, for large graphs, a multilevel approach [2, 10, 13, 27] can provide an equally good global view, while being much faster because the calculation of the spectral vector of a large matrix is avoided. A number of efficient and high-quality graph partitioning codes based on the multilevel approach have been developed [10, 13, 27]. The success of the multilevel approach in graph partitioning motivated the work reported in this paper.

We propose a multilevel algorithm for the ordering of sparse symmetric matrices. Numerical tests on a wide range of problems from different application areas confirm that our multilevel algorithm yields orderings of comparable quality to the Hybrid algorithm. Moreover, our algorithm does not require any spectral information and is significantly faster than the Hybrid algorithm. So far as the authors know, the only other attempt at using a multilevel approach in this context has been by Bolman and Hendrickson [3]. Their algorithm combined a multilevel approach with a 1-sum local refinement procedure. However, they were not able to consistently improve on the quality of the original Sloan algorithm.

This paper is organised as follows. In Section 2, definitions and terminology are introduced, and the Sloan and Hybrid algorithms are recalled. In Section 3, our multilevel approach is presented. Numerical results comparing our method with the Sloan and the Hybrid algorithms are given in Section 4. Section 5 summarizes our findings and considers future directions for research.

2 Background

2.1 Definitions

We first need to introduce some nomenclature and notation. Let $A = \{a_{ij}\}$ be an $n \times n$ symmetric matrix. At the i-th step of the factorization of A, row k is said to be *active* if $k \geq i$ and there exists a column index $l \leq i$ such that $a_{kl} \neq 0$. The i-th *wavefront* f_i of A is defined to be the number of rows that are active during the i-th step of the factorization. The *maximum* and *root-mean-squared* (RMS) wavefronts are, respectively,

$$F(A) = \max_{1 \leq i \leq n} \{f_i\}$$

and

$$rmsf(A) = \left(\frac{\sum_{i=1}^{n} f_i^2}{n} \right)^{\frac{1}{2}}.$$

The *profile* of A is the total number of entries in the lower triangle when any zero ahead of the first entry in its row is excluded, that is,

$$P(A) = \sum_{i=1}^{n} \max_{a_{ij} \neq 0} \{i + 1 - j\}. \quad (1)$$

It is straightforward to show that

$$P(A) = \sum_{i=1}^{n} f_i.$$

For a frontal solver these statistics are important because

- the memory needed to store the frontal matrix is F^2
- P is the total storage needed for the factorised matrix
- the number of floating-point operations when eliminating a variable is proportional to the square of the current wavefront size.

Our goal therefore is to construct an efficient ordering algorithm that reduces the above quantities.

It is often convenient when developing ordering algorithms to treat the matrix A in terms of its adjacency graph. An undirected graph $\mathcal{G}$ is defined to be a pair (V, E), where V is a finite set of *vertices* (or *nodes*) and E is a finite set of *edges* defined as unordered pairs of distinct vertices. In a *weighted* graph, each vertex and edge has a weight associated with it. The adjacency graph $\mathcal{G}(A)$ of the square symmetric matrix A comprises the vertices $V(A) = \{1, 2, \ldots, n\}$ and the edges

$$E(A) = \{(i, j) \mid a_{ij} \neq 0,\ i > j\}.$$

Two vertices i and j are said to be *neighbours* (or to be *adjacent*) if they are connected by an edge. The notation $i \leftrightarrow j$ will be used to show that i and j are neighbours. The *adjacency* set for i is the set of its neighbours, that is,

$$adj(i) = \{j \mid j \leftrightarrow i,\ i, j \in V\}.$$

The *degree* of $i \in V$ is $deg(i) = |adj(i)|$, the number of neighbours. If X is a subset of V, its adjacency set is defined to be

$$adj(X) = \cup_{j \in X} adj(j) \setminus X.$$

Observe that, given the graph representation of a symmetric matrix, the i-th wavefront can be defined as the vertex i plus the set of vertices adjacent to the vertex set $\{1, 2, \ldots, i\}$, that is,

$$f_i = adj\left(\{1, 2, \ldots, i\}\right) \cup \{i\}.$$

A *path* of length k in $\mathcal{G}$ is an ordered set of distinct vertices $\{v_1, v_2, \ldots, v_k, v_{k+1}\}$, with $v_i \leftrightarrow v_{i+1}$ $(1 \leq i \leq k)$. Two vertices are *connected* if there exists a path between them. A graph $\mathcal{G}$ is connected if each pair of distinct vertices is connected. The *distance*, $dist(u, v)$, between two vertices u and v in $\mathcal{G}$ is the length of the shortest path connecting them. The *eccentricity* of a vertex u is defined to be

$$e(u) = \max\{dist(u, v) \mid v \in \mathcal{G}\}.$$

The *diameter* of $\mathcal{G}$ is then

$$\delta(\mathcal{G}) = \max\{e(u) \mid u \in \mathcal{G}\}.$$

A vertex u is a *peripheral* vertex if its eccentricity is equal to the diameter of the graph, that is, $e(u) = \delta(\mathcal{G})$. A *pseudoperipheral* vertex u is defined by the condition that, if v is any vertex for which $dist(u, v) = e(u)$, then $e(v) = e(u)$. The pair u, v of pseudoperipheral vertices define a *pseudodiameter*.

Throughout our discussion, it is assumed that the matrix A of interest is irreducible so that its adjacency graph $\mathcal{G}(A)$ is connected. Disconnected graphs can be treated by considering each component separately.

2.2 The Sloan algorithm

The Sloan algorithm [26], as presented in [17, 22], orders the vertices of a weighted adjacency graph. The weighted graph is derived from the unweighted graph by "condensing" vertices to form supervertices. Vertices i and j are condensed into a supervariable if

$$i \cup adj(i) = j \cup adj(j). \tag{2}$$

The weight of a supervertex is the number of unweighted vertices it represents. The use of condensing can sometimes reduce the size of the graph considerably, thus reducing the time required for reordering [5, 22]. This is particularly true for matrices arising from finite-element calculations where it is common for there to be several degrees of freedom at each node in the finite-element grid.

The Sloan algorithm for reordering a graph has two distinctive phases:

1. selection of a start vertex s and an end vertex e
2. vertex reordering.

Step 1 looks for a pseudodiameter of the weighted graph and chooses s and e to be the endpoints of the pseudodiameter. A pseudodiameter may be computed using a modification of the Gibbs-Poole-Stockmeyer algorithm (see [22]). In Step 2, the pseudodiameter is used to guide the reordering. Each vertex of the weighted graph is given a priority $P(i)$ where

$$P(i) = -W_1 \; inc(i) + W_2 \; dist(i, e) \tag{3}$$

and (W_1, W_2) are positive weights. The first term, $inc(i)$, is the amount by which the wavefront will increase if vertex i is ordered next. The second term, $dist(i, e)$, is the distance between i and the end vertex e. The start vertex s is ordered first then, at each stage, the next vertex is chosen among eligible vertices with the highest priority. Thus, a balance is maintained between the aim of keeping the wavefront small and bringing in vertices that have been left behind (far away from e). The list of eligible vertices comprise those that are in the front (neighbours of one or more renumbered vertices) or neighbours of one or more vertices in the front. The best choice for the weights (W_1, W_2) is problem dependent. Based on experimentation, Reid and Scott [22] suggest that the two pairs of weights $(2, 1)$ and $(16, 1)$ should be tried and the better ordering chosen. By default, when implementing the Sloan algorithm, the code `MC60` tries both these pairs of weights but the user may also choose to select other weights.

Once an ordering for the vertices of the weighted graph has been obtained, an ordering for A can be constructed.

2.3 The Hybrid algorithm

The first term in (3) affects the priority function in a local way, by giving higher priority to vertices that will result in a small (or negative) increase to the current wavefront. This is done in a greedy fashion, without consideration of the long-term effect. The second term acts in a more global manner, ensuring vertices lying far away from the end vertex are not left behind. Step 2 of the Sloan algorithm can therefore be viewed as an algorithm that refines the ordering implied by the distance function $dist(i, e)$.

The distance function in (3) can be replaced by other orderings that provide a global view. In particular, the spectral ordering may be used. The spectral algorithm associates a Laplacian matrix $L = \{l_{ij}\}$ with the symmetric matrix A as follows:

$$l_{ij} = \begin{cases} -1, & \text{if } \; i \neq j \text{ and } \; i \leftrightarrow j, \\ deg(i), & \text{if } \; i = j, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

An eigenvector corresponding to the smallest positive eigenvalue of the Laplacian matrix is called a *Fiedler vector.* The spectral algorithm orders the vertices of $\mathcal{G}(A)$ by sorting the components of the Fiedler vector into monotonic order. This approach has been found to produce small profiles and wavefronts [1].

The spectral algorithm has been used in the context of graph partitioning [9], where it has been found that results can be improved by incorporating a local refinement step. This refinement performs local optimizations and smoothes out local oscillations that may be present. In the context of wavefront reduction, the Hybrid algorithm of Kumfert and Pothen [17] combines the spectral algorithm with Step 2 of the Sloan algorithm. In our view, it is this combination of global and local ordering algorithms that accounts for the good performance of the Hybrid algorithm, particularly for very large problems.

The Hybrid algorithm, as presented in [22], chooses as the start vertex s the first vertex in the spectral ordering and replaces (3) with the priority function

$$P(i) = -W_1\ inc(i) - W_2\ \nu\ p(i). \tag{5}$$

Here ν is a normalising factor and $p(i)$ is the position of vertex i in the spectral ordering, also referred to as its global priority value. ν is chosen so that the factor for W_2 varies up to $dist(s,e)$, as in (3) (see [22]). On the basis of their numerical experimentation, Reid and Scott [22] propose the pairs of weights $(1,2)$ and $(16,1)$. Note that $(1,2)$ is recommended instead of the pair $(2,1)$ used by the Sloan algorithm. Reid and Scott argue that the global priority based on the spectral ordering has been found to be better than that obtained from a pseudodiameter, justifying a larger value for W_2 in this case. Numerical experiments have shown that, for large problems, the Hybrid method can significantly outperform the original Sloan algorithm, although it requires significantly more CPU time. This is illustrated in Section 4.

3 The Multilevel Ordering Algorithm

The matrix ordering algorithm proposed in this paper is based on a multilevel approach. Given the adjacency graph $\mathcal{G}(A)$, a series of graphs are generated, each coarser than the preceding one. The coarsest graph is then ordered. This ordering is recursively prolonged to the next finer graph, local refinement is performed at each level, and the final ordering on the finest graph gives an ordering for A.

3.1 The Multilevel Approach

In the context of graph partitioning, the multilevel approach generates a series of coarser and coarser graphs [2, 10, 14, 27]. The aim is for each successive graph to encapsulate the information needed to partition its "parent", while containing fewer vertices and edges. The coarsening continues until a graph with only a small number of vertices is reached. This can be partitioned cheaply. The partitions on the coarse graphs are recursively prolonged (usually by injection) to the finer graphs, with further refinement at each level.

One of the first uses of a multilevel approach for the partitioning of undirected graphs was reported by Barnard, Pothen and Simon [2]. Motivated by the need to reduce the time for computing the Fiedler vector, Barnard *et al.* combined a multilevel approach with a spectral bisection algorithm. It was soon realized [10, 15, 27] that the multilevel approach provides a global view of the problem, and therefore can be used to advantage with a good local optimizer. In graph partitioning, the Kernighan-Lin algorithm [16] is used and, combined with the multilevel approach, has proved very successful at rapidly computing high quality partitions.

Given the success of the multilevel approach for graph partitioning, it is perhaps surprising that, as far as the authors are aware, there has only been one reported attempt at applying it to the problem of profile and wavefront reduction. Bolman and Hendrickson [3] introduce a *weighted 1-sum* metric

$$\sigma_1(A) = \sum_{i=1}^{n} \sum_{j \leftrightarrow i,\ j<i} w_{ij}(i-j),$$

where w_{ij} are edge weights, and aim to minimise $\sigma_1(A)$. The edge weights are all one on the finest graph; on the coarser graphs, edge weights are assigned as edges are collapsed (see Section 3.1.1 below). Although Bowman and Hendrickson's objective is to minimise the matrix envelope

$$Env(A) = \sum_{i=1}^{n} \max_{j \leftrightarrow i,\ j<i} \{i-j\},$$

(which is equal to $P(A) - n$, where $P(A)$ is the profile (1) of A), they report that they found it more efficient and effective to work with the 1-sum. Bowman and Hendrickson propose combining a multilevel approach with a refinement algorithm that is similar to Kernighan-Lin and is based on swapping consecutive vertices. The gain in swapping each such pair of vertices k and $k+1$ is calculated initially and then updated during the refinement. The gain from a swap is measured using the weighted 1-sum. Numerical results presented in [3] are not very encouraging because, as well as being slower than the Sloan algorithm, the quality of the ordering produced by the Bowman and Hendrickson multilevel algorithm is often poorer.

Our view is that, because the reordering phase of the Sloan algorithm provides a good local refinement algorithm, it should be used directly with the multilevel approach, and this is the basis of our new multilevel wavefront reduction algorithm. Our proposed algorithm has three distinct phases: coarsening, coarsest graph ordering, and, finally, prolongation and refinement. We discuss each of these phases before outlining our proposed algorithm.

3.1.1 The coarsening phase

There are a number of ways to coarsen an undirected graph, two of which are briefly discussed here. The most popular method is based on edge collapsing [10, 15], in which pairs of adjacent vertices are selected and each pair is coalesced into one new vertex. Each vertex of the resulting coarse graph has an associated weight, equal to the number of original vertices it represents. Each edge of the coarse graph also has a weight associated with it. Initially, all edge weights are

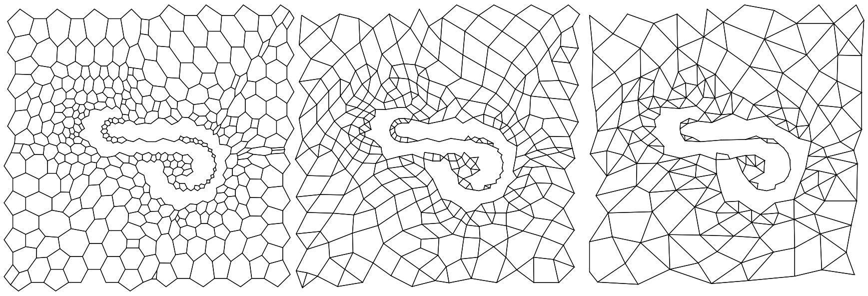

Figure 1: Graph coarsening by edge collapsing: the original graph $\mathcal{G}$ with 788 vertices (left); $\mathcal{G}_1$ with 403 vertices (centre); $\mathcal{G}_2$ with 210 vertices (right).

set to one. During coarsening, edge weights are unchanged unless both merged vertices are adjacent to the same neighbour. In this case, the new edge is given a weight equal to the sum of the weights of the edges it replaces. The edges to be collapsed are usually selected using a *maximal matching*. This is a maximal set of edges, no two of which are incident to the same vertex.

For undirected graph partitioning, *heavy-edge matching* [15] has been found to work well. Here, the idea is to preferentially collapse heavier edges. When looking through a neighbour list for an unmatched vertex, an edge with the largest weight is selected. Heavy-edge matching has the advantage that the total weight of the edges of the resulting coarse graph is relatively small, and consequently its partitioning is more likely to give a small edge-cut (the sum of the weights of the edges cut by the partitioning). Additionally, if the partitioning on the coarsest graph is injected to the finer graphs without further refinement, the edge-cut on the coarsest graph equals that on the finest (original) graph. This fact, hereafter referred to as the *inheritance property*, results in the original problem being encapsulated well by the coarser problems. Figure 1 illustrates a graph $\mathcal{G}$ with 788 vertices, together with 2 levels of coarsening using edge collapsing, giving graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ with 403 and 210 vertices, respectively.

Other coarsening methods have been proposed. In [2], a *maximal independent vertex set* of a graph is chosen as the vertices for the coarse graph. An independent set of vertices is a subset of the vertices such that no two vertices in the subset are connected by an edge in the graph. An independent set is maximal if the addition of an extra vertex always destroys the independence. An algorithm for constructing a maximal independent set is discussed in Section 4.2. Edges of the coarse graph are formed through a process based on the Galerkin product (see Section 3.1.4 for details), which effectively links two vertices in the maximal independent vertex set by an edge if their distance apart is no greater than three. Figure 2 illustrates the same original graph $\mathcal{G}$ with 788 vertices, together with 2 levels of coarsening using this method, giving graphs with 332 and 94 vertices, respectively.

In Section 4, the results of using both edge collapsing and the maximal independent vertex set for coarsening within our wavefront reduction algorithm are presented.

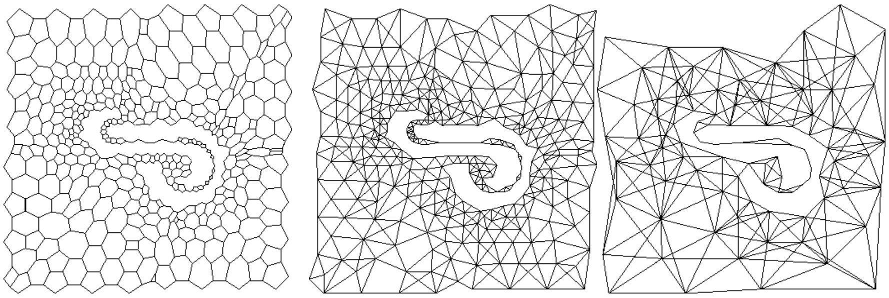

Figure 2: Graph coarsening based on the maximal independent vertex set: the original graph $\mathcal{G}$ with 788 vertices (left); $\mathcal{G}_1$ with 332 vertices (centre); $\mathcal{G}_2$ with 94 vertices (right).

The chosen coarsening process is applied recursively until one of the following is achieved:

- the number of levels exceeds a preset limit
- the number of vertices in the coarsest graph is less than a preset number (chosen to be 100 in this study)
- the ratio of the number of vertices in two successive graphs exceeds a preset constant (0.8 in this study).

The last condition is necessary, particularly if edge collapsing is used, for the following reasons. After a number of levels of coarsening it is possible that the coarsest graph has one supervertex with a very high vertex weight, possibly exceeding 50% of the total. In this case, subsequent coarsening will not reduce the size of the graph significantly. Furthermore, a multilevel algorithm with only a small reduction between fine and coarse graph sizes will have a high algorithmic complexity. We therefore stop coarsening if the ratio of the number of vertices in successive graphs is greater than 0.8.

3.1.2 The coarsest graph ordering

Because the coarsest graph has a small number of vertices and edges, it can be reordered quickly using any standard profile reduction algorithm. We have used both the Sloan and the Hybrid algorithms and present results for both approaches in Section 4.

3.1.3 The prolongation and refinement phase

During the prolongation phase, the vertices of the fine graph are given global priority values by mapping the coarse graph ordering onto the fine graph. This mapping can be represented by a prolongation matrix P. If the coarse and fine graphs have n_c and n_f vertices, respectively, the prolongation matrix is of order $n_f \times n_c$.

When coarsening is by edge collapsing, the position of a vertex j in the coarse graph ordering is injected to give the global priority values of its parent (or parents). A parent of j is defined to be a vertex on the fine graph that either coalesces into j, or remains as j itself. The prolongation matrix has entries p_{ij} given by

$$p_{ij} = \begin{cases} 1, & \text{if fine graph vertex } i \text{ is a parent of coarse graph vertex } j, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

When coarsening is based on a maximal independent vertex set, the coarse graph vertices comprise the maximal independent set of the fine graph. The global priority value of a vertex in the fine graph that belongs to the maximal independent set is defined as the position of this vertex in the coarse graph ordering. The global priority value of a fine graph vertex not in the maximal independent set is calculated by averaging the global priority values of its neighbours that belong to the maximal independent set (by definition there is at least one such neighbour). For each coarse graph vertex j, let $fine(j)$ denote the corresponding fine graph vertex. For each fine graph vertex i, define $mdeg(i)$ to be the number of neighbouring fine graph vertices that belong to the maximal independent set. The prolongation matrix has entries

$$p_{ij} = \begin{cases} 1, & \text{if } i = fine(j), \\ \frac{1}{mdeg(i)}, & \text{if } i \leftrightarrow fine(j),\ i \neq fine(j), \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

The global priority values are refined using Step 2 of the Sloan algorithm (that is, by using the priority function (5)) to give the final ordering for the fine graph.

3.1.4 The multilevel algorithm

We now formulate our multilevel wavefront reduction algorithm. For this it is convenient to introduce some further notation. The subscripts f and c are used to represent fine and coarse graph quantities, respectively. For example, $\mathcal{G}_f$ denotes the fine graph with n_f vertices and $\mathcal{G}_c$ is the graph with n_c vertices obtained after coarsening ($n_c < n_f$). We will associate with $\mathcal{G}_f$ an $n_f \times n_f$ matrix G_f which has zero diagonal entries and nonzero off-diagonal entries e_{ij} if and only if vertices i and j are adjacent in $\mathcal{G}_f$ with edge weight e_{ij}. G_c is defined analogously.

If P denotes an $n_f \times n_c$ prolongation matrix, the coarse graph may expressed as the Galerkin product

$$G_c \leftarrow P^T\, G_f\, P.$$

This expression means that the matrix product $P^T G_f P$ is computed and the matrix G_c is obtained by setting the diagonal entries of the resulting matrix to zero.

The *global priority vector* p of a graph is a vector with entries $p(i)$, where $p(i)$ is the global priority value of vertex i. This vector indicates the preferred ordering of the vertices. For the Hybrid algorithm, the global priority vector is obtained by ordering the vertices based on the values of the Fiedler vector. Note that the global priority vector need **not** be a permutation vector. We let $CoarsestOrder(\mathcal{G}, w)$ be an algorithm that takes the coarsest graph $\mathcal{G}$ with associated vertex weights w, and returns an ordering for $\mathcal{G}$. Furthermore, we let $SloanRefine(\mathcal{G}, w, p^0)$ denote the algorithm that takes the graph $\mathcal{G}$, its vertex weights w, and a global priority vector p^0, and returns a refined ordering for $\mathcal{G}$ using (5) with $p = p^0$.

With this notation, if $MinSize$ is the preset number of vertices beyond which there is no further coarsening, our multilevel wavefront reduction algorithm can be formally presented as follows. The starting point is the fine graph $\mathcal{G}_f$ with vertex weights w_f and associated matrix G_f.

function MultilevelOrder $(\mathcal{G}_f,\ w_f)$

- *If* $n_f \ < \ MinSize$ *then*
  - $p_f \ = CoarsestOrder(\mathcal{G}_f,\ w_f)$
  - *return* p_f
- *The coarsening phase:*
  - *set up the* $n_f \times n_c$ *prolongation matrix* P
  - $G_c \ \leftarrow \ P^T \ G_f \ P$
  - $w_c \ = \ P^T \ w_f$
  - $p_c \ = \ MultilevelOrder\ (\mathcal{G}_c,\ w_c)$
- *The prolongation and refinement phase:*
  - $p_f^0 \ = \ P \ p_c$
  - $p_f \ = \ SloanRefine(\mathcal{G}_f, w_f, p_f^0)$
  - *return* p_f

Figure 3 illustrates the multilevel algorithm applied to the test problem bcsstk11 (see Appendix). A maximal independent vertex set is used for coarsening (see Section 4.2 for details). Notice that on the coarsest level, the lower right-hand part of the matrix after reordering (bottom right) has no nonzero entries. This is because the graph corresponding to the bcsstk11 matrix has nine components, eight of which are small, and coarsening gives eight isolated vertices on the coarsest level. Since diagonal elements are not displayed, after reordering there is an 8×8 null matrix in the lower right hand part of the coarsest matrix.

The two-level ordering algorithm using heavy-edge coarsening is illustrated in Figure 4. Edges $(v1, v2)$, $(v3, v4)$ and $(v5, v6)$ are collapsed to give the coarse graph vertices $u1$, $u2$ and $u3$. Based on (6), the prolongation matrix is

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}.$$

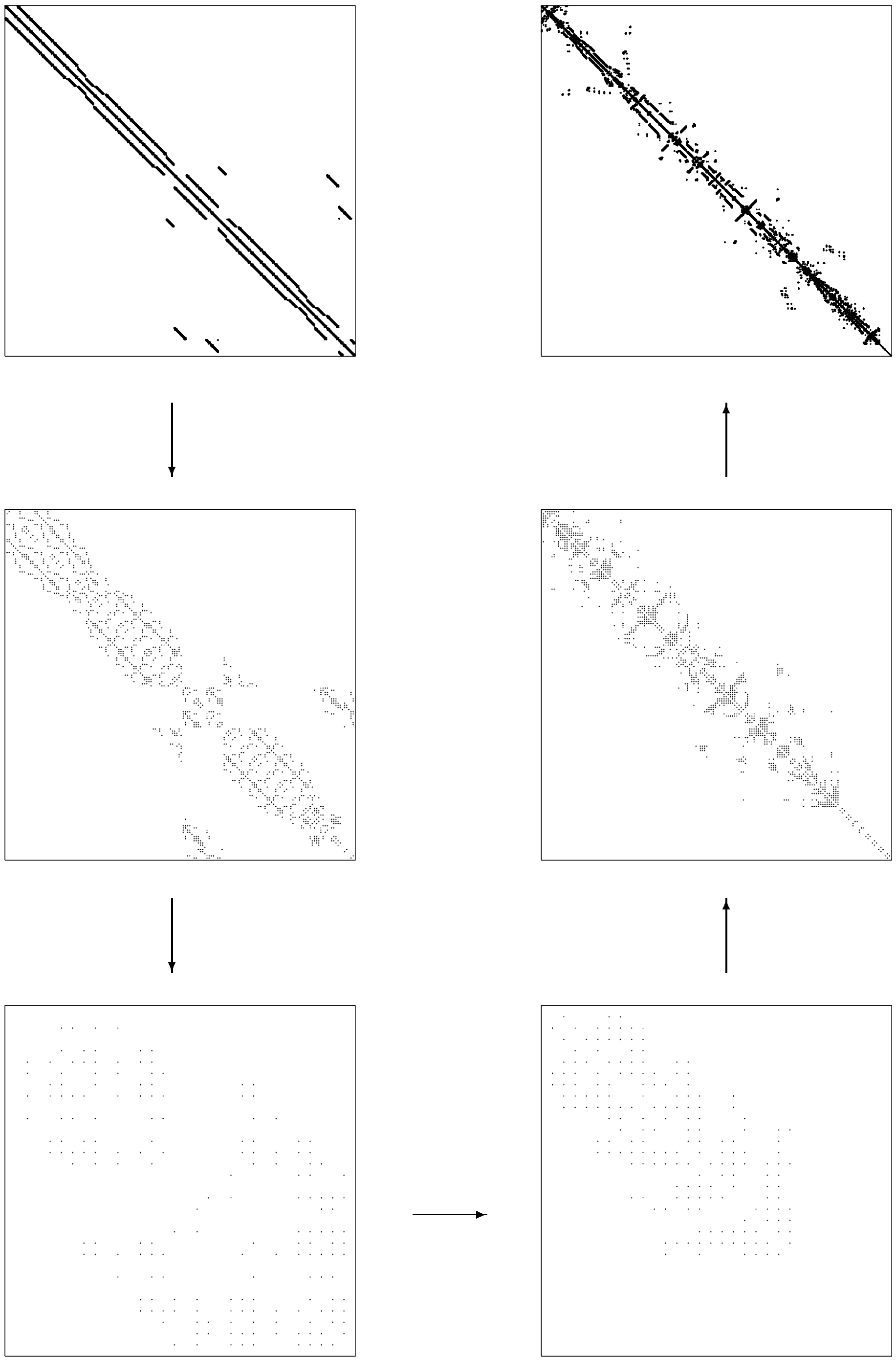

Figure 3: An illustration of the multilevel algorithm using problem bcsstk11. The original matrix (top left) of order 1473 is coarsened twice to give the coarse matrices on the left (of order 162 and 30, respectively). The coarsest matrix is ordered (bottom right), and prolongated and refined to give the final ordering (top right) for the original matrix. The multilevel algorithm gives a RMS wavefront of 46.55, which is smaller than that given by the Sloan algorithm (51.40) and the Hybrid algorithm (47.76), and significantly smaller than the RMS wavefront of the original matrix (104.34).

With the edge weights chosen as in the figure, it follows that

$$P^T G_f P = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}^T \begin{pmatrix} 0 & 5 & 2 & 0 & 0 & 0 \\ 5 & 0 & 2 & 0 & 1 & 2 \\ 2 & 2 & 0 & 4 & 1 & 0 \\ 0 & 0 & 4 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 3 \\ 0 & 2 & 0 & 0 & 3 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 10 & 4 & 3 \\ 4 & 8 & 2 \\ 3 & 2 & 6 \end{pmatrix}.$$

Thus, setting the diagonal entries to zero, we have

$$G_c = \begin{pmatrix} 0 & 4 & 3 \\ 4 & 0 & 2 \\ 3 & 2 & 0 \end{pmatrix}.$$

The edge weights for the three edges of the coarse graph are 4, 3 and 2 respectively, as shown in Figure 4 (middle). The vertex weights for the fine graph (top left of the figure) are chosen to be $w_f = (2,1,4,3,1,3)^T$. The vertex weights for the coarse graph are therefore

$$w_c = P^T w_f = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}^T \begin{pmatrix} 2 \\ 1 \\ 4 \\ 3 \\ 1 \\ 3 \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \\ 7 \end{pmatrix}.$$

Assuming the coarse graph is ordered as $p_c(u1) = 1$, $p_c(u2) = 2$ and $p_c(u3) = 3$, the global priority vector for the fine graph is

$$p_f^0 \;=\; P\, p_c = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 2 \\ 2 \\ 3 \\ 3 \end{pmatrix}.$$

This is shown on the top right-hand part of Figure 4.

Figure 5 illustrates the two-level ordering algorithm using a maximal independent vertex set. The vertices $v1$, $v4$ and $v6$ are chosen to form the maximal independent set. Vertex $v2$ has two neighbours in the maximal independent set ($v1$, $v6$), therefore $mdeg(v2) = 2$. Similarly, $mdeg(v3) = mdeg(v5) = 2$. Thus according to (7), the prolongation matrix is

$$P = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{pmatrix}.$$

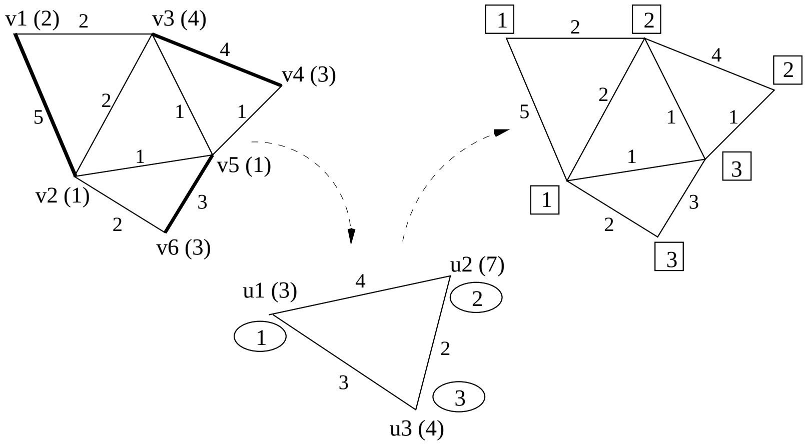

Figure 4: A graph (top left) is coarsened by heavy-edge collapsing to give the coarse graph (middle). The coarse graph is ordered and the ordering is prolonged to give the priority vector for the fine graph (top right). Numbers in parentheses are vertex weights, numbers in ellipses are the ordering, numbers in squares are the global priority values and numbers along the edges are edge weights.

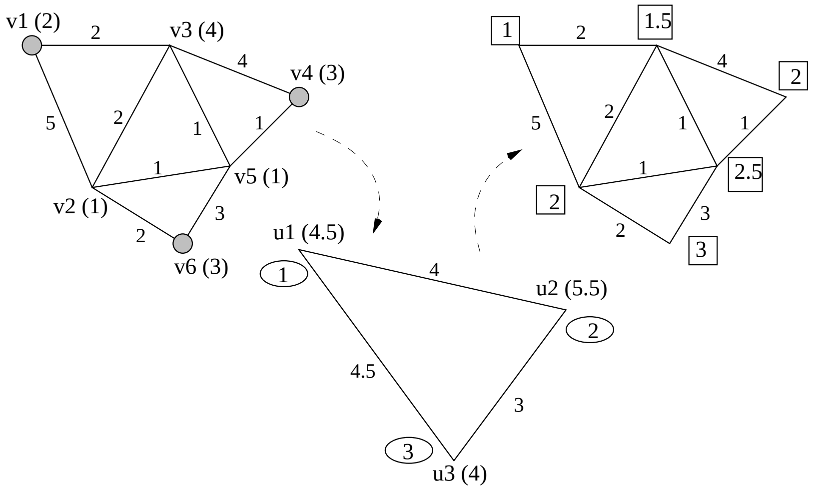

Figure 5: A graph (top left) is coarsened using the maximal independent vertex set (shaded and circled) to give the coarse graph (middle). This coarse graph is ordered and the ordering is prolonged to give the priority vector for the fine graph (top right). Numbers in brackets are vertex weights, numbers in ellipses are the ordering, numbers in squares are the global priority values and numbers along the edges are edge weights.

It follows that the Galerkin product is

$$P^T G_f P = \frac{1}{4}\begin{pmatrix} 2 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 2 \end{pmatrix}^T \begin{pmatrix} 0 & 5 & 2 & 0 & 0 & 0 \\ 5 & 0 & 2 & 0 & 1 & 2 \\ 2 & 2 & 0 & 4 & 1 & 0 \\ 0 & 0 & 4 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 3 \\ 0 & 2 & 0 & 0 & 3 & 0 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 2 \end{pmatrix} = \begin{pmatrix} 8 & 4 & 4.5 \\ 4 & 5.5 & 3 \\ 4.5 & 3 & 5.5 \end{pmatrix}.$$

Setting the diagonal to zero gives the coarse graph matrix

$$G_c = \begin{pmatrix} 0 & 4 & 4.5 \\ 4 & 0 & 3 \\ 4.5 & 3 & 0 \end{pmatrix}.$$

The edge weights for the three edges of the coarse graph are 4, 4.5 and 3 respectively, as shown in Figure 5 (middle). The vertex weights for the fine graph again taken to be $w_f = (2,1,4,3,1,3)^T$, the vertex weights for the coarse graph are given by

$$w_c = P^T w_f = \frac{1}{2}\begin{pmatrix} 2 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 2 \end{pmatrix}^T \begin{pmatrix} 2 \\ 1 \\ 4 \\ 3 \\ 1 \\ 3 \end{pmatrix} = \begin{pmatrix} 4.5 \\ 5.5 \\ 4 \end{pmatrix}. \tag{8}$$

Assuming again that the coarse graph is ordered as $p_c(u1) = 1$, $p_c(u2) = 2$ and $p_c(u3) = 3$, the global priority vector for the fine graph is

$$p_f^0 \;=\; Pp_c = \frac{1}{2}\begin{pmatrix} 2 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 1.5 \\ 2 \\ 2.5 \\ 3 \end{pmatrix}. \tag{9}$$

This is shown on the right-hand side of Figure 5.

4 Numerical results

In this section, our multilevel approach is compared with the Sloan and Hybrid algorithms on a large set of test problems. All codes are written in Fortran and were developed with a view to efficient implementation. The experiments are performed on a COMPAQ computer with a 300 MHz Alpha EV5 processor.

The `MC60` code of Reid and Scott [22] is used in the experiments for the Sloan algorithm, for ordering the coarsest graph, and for subsequent refinement. We experimented with using vertex weights when calling `MC60` at each of the levels, but found that it led to a deterioration in the quality of the orderings (see Section 4.4 for further details). Therefore, unless specified otherwise, we set the vertex weights equal to one when calling `MC60`. Edge weights are not used in this paper.

The spectral ordering needed for the Hybrid algorithm is computed using a multilevel Fiedler vector code written by the first author. This code implements the algorithm as described in [2].

We take the coarse graph ordering algorithm $CoarsestOrder$ to be either the Sloan or the Hybrid algorithm. We denote by $Sloan(S,K)$ (and $Hybrid(S,K)$) the Sloan (respectively, Hybrid) algorithm on the coarsest graph and with coarsening scheme S on up to K levels. Thus $Sloan(S,1)$ is the standard Sloan algorithm and $Hybrid(S,1)$ the Hybrid algorithm. Coarsening schemes based on edge collapsing and maximal independent vertex set are denoted by $S = EC$ and $S = MIV$, respectively. Thus $Sloan(EC,3)$ is the multilevel algorithm with edge collapsing and a maximum of 3 levels, and the Sloan algorithm used on the coarsest graph.

Our suite of 101 test problems is listed in alphabetical order in the Appendix. The problems are all symmetric, and range in order from 66 (dwt66) to 224,617 (Halfb). The problems come from a variety of practical applications and have been taken from the Harwell-Boeing Sparse Matrix Collection (`http://www.cse.clrc.ac.uk/Activity/SparseMatrices`), MatrixMarket (`http://math.nist.gov/MatrixMarket`), and the test set of Kumfert and Pothen [17], with additional finite element problems supplied by Christian Damhaug of Det Norske Veritas, Norway. Also given in the Appendix are the initial RMS wavefronts ($rmsf$) for each matrix and the ratio, ρ, between the RMS wavefronts before and after reordering with the Hybrid algorithm. In general, the Hybrid algorithm substantially improves the ordering (although there are a small number of exceptions, notably problems bscctk13, bcsstk31, and sstmodel).

4.1 Multilevel algorithm with coarsening by edge collapsing

Because of its success in graph partitioning, coarsening by edge collapsing was the first strategy we tried. Figure 6 compares the RMS wavefront for the Sloan, Hybrid, and multilevel algorithms. For the multilevel algorithm, the Sloan algorithm is used to order the coarsest graph (results for the Hybrid algorithm on the coarsest graph are included in Section 4.3). In this and subsequent figures, comparisons are given with respect to the Hybrid algorithm so that the RMS wavefront for each algorithm is divided by the corresponding RMS wavefront for the Hybrid algorithm, and averaged over the test cases to give a relative score for the algorithm. The smaller the score, the better the algorithm. With this metric, the Hybrid algorithm always has a score of one. To show the effect of matrix order, the scores for each algorithm for matrices of order greater than $37 \times 3^k (1 \leq k \leq 8)$ are plotted separately in the figure, with the number of matrices over the threshold printed in brackets. A log scale is used for the x axis (matrix order).

A number of interesting features can be observed. The first observation

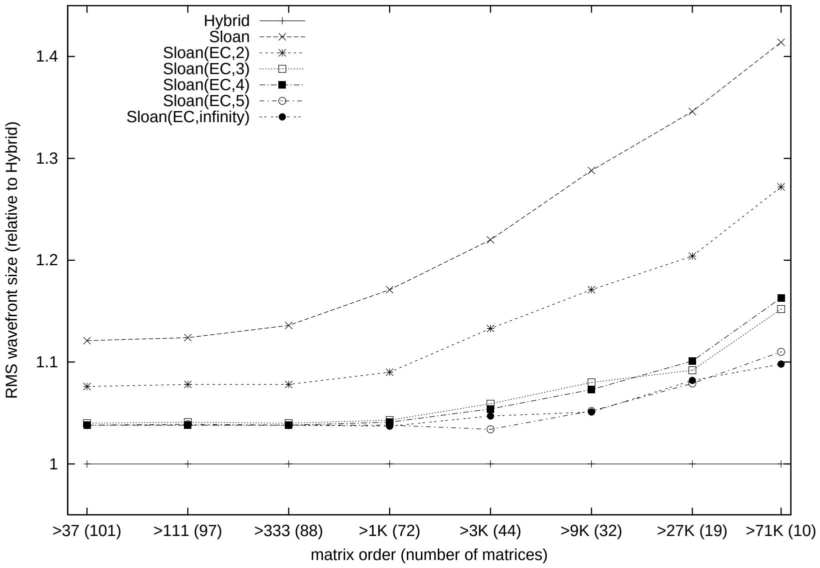

Figure 6: A comparison the RMS wavefronts of the Sloan, the Hybrid and the $Sloan(EC, K)$ algorithms.

is that, relative to the Hybrid algorithm, the RMS wavefront given by the Sloan algorithm deteriorates as the order of the matrix increases. Overall, the RMS wavefront for the Sloan algorithm is about 12% greater than for the Hybrid algorithm, whilst for the largest 10 matrices, it is about 41% more. This deterioration further confirms the lack of a global view of the Sloan algorithm and supports the earlier findings reported in [17, 22].

The second observation is that, as the number of levels K in the multilevel $Sloan(EC, K)$ algorithm increases, the RMS wavefront decreases. The multilevel algorithm without a preset limit for the maximum number of levels, $Sloan(EC, \infty)$, is significantly better than the Sloan algorithm. However, it does not perform quite as well as the Hybrid algorithm, producing RMS wavefronts that are on average 5% larger (10% larger for the 10 largest problems).

This performance of the multilevel algorithm based on edge collapsing is perhaps not entirely unexpected. As already discussed, for graph partitioning, the edge collapsing strategy has the inheritance property. This is not the case for wavefront ordering. The wavefront for a given ordering of a coarse graph is unlikely to be the same as that for the corresponding fine graph. Although this does not exclude the usefulness of the multilevel approach for the wavefront reduction problem, it does remove the special preference for edge collapsing based coarsening over other coarsening methods.

We also note that, for coarsening by edge collapsing, a pseudodiameter of the coarse graph need not correspond well to a pseudodiameter of the original graph. This will be the case if the graph is coarsened along one direction, as illustrated in Figure 7. Here the coarse graph start and end vertices S_c, E_c are

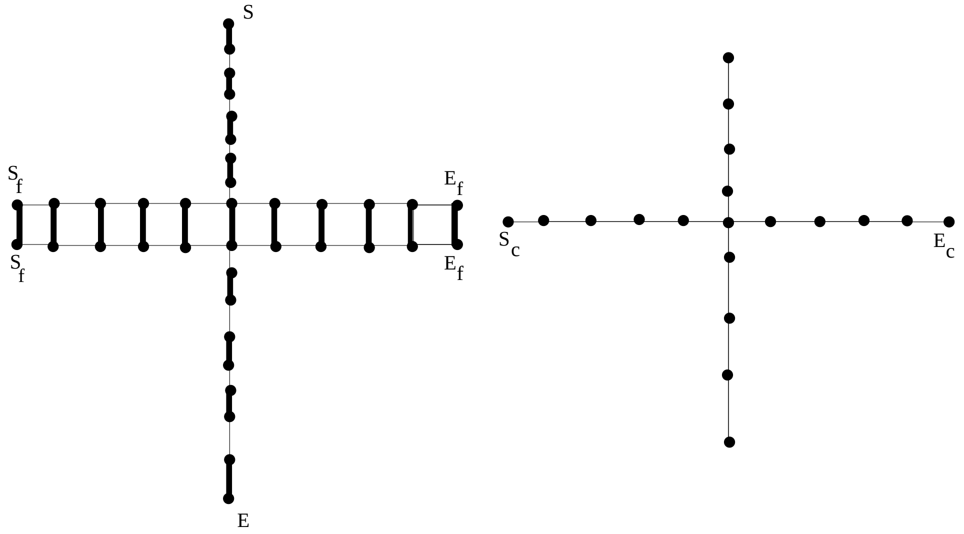

Figure 7: The (fine) graph on the left and its coarsened graph on the right. The heavy edges are collapsed. The modified Gibbs-Poole-Stockmeyer algorithm of Reid and Scott [22] applied to the fine graph would give the start and end vertices S and E. On the coarse graph (right), it would give S_c and E_c, when injected to the fine graph the start and end vertices become S_f and E_f.

far from the fine graph start and end vertices S, E. Although it is difficult to tell whether this directional coarsening occurs in our test problems, the fact that it can happen is of concern.

In terms of CPU time (Figure 8), the $Sloan(EC, K)$ algorithm is about 40% faster than the hybrid algorithm. The Sloan algorithm is more than four times faster than the Hybrid algorithm. This is in line with an observation made by Kumfert and Pothen [17], where a factor of five is reported.

4.2 Multilevel algorithm with coarsening based on the maximal independent vertex set

Although coarsening by edge collapsing has been popular and successful in multilevel algorithms for graph partitioning, results in Section 4.1 show that for wavefront reduction, this approach improves on the Sloan algorithm but is not as good as the Hybrid algorithm. Directional coarsening is a possible reason.

When used for the multilevel spectral algorithm, a maximal independent set is chosen in a greedy fashion by picking unmatched vertices at random and, when a vertex is picked, masking its neighbouring vertices as matched [1, 2]. We, however, adopt a more sophisticated algorithm that both yields slightly improved orderings and requires less CPU time (see Section 4.4 for a more detailed discussion). Our algorithm is based on that of Ruge and Stuben [23], which has been used successfully in the field of algebraic multigrid. This algorithm was designed for unsymmetric matrices; we have modified it for

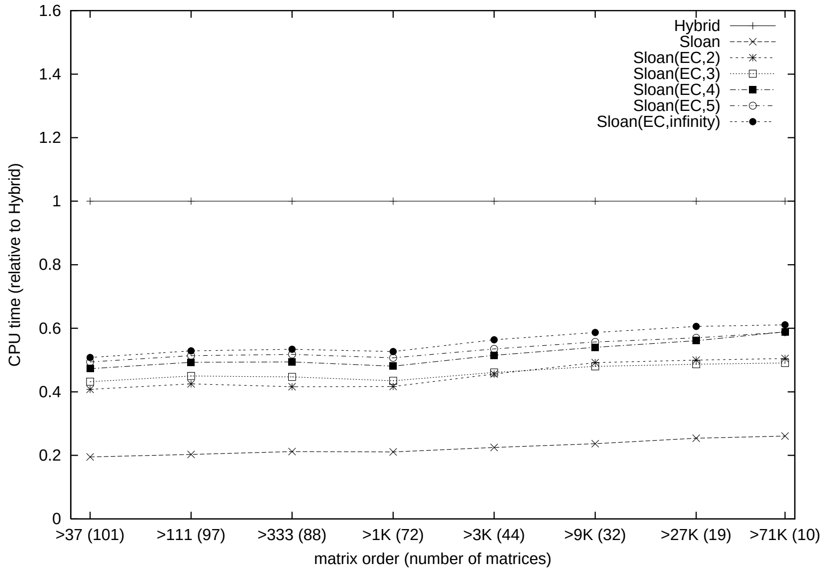

Figure 8: A comparison of the CPU times for the Sloan, the Hybrid and the $Sloan(EC, K)$ algorithms.

symmetric matrices. At each step, each vertex in V lies in one of three sets: it is either uncoloured (V_U), or is in the maximal independent set (V_C), or is not a candidate for the maximal independent set (V_F). Each vertex has a gain value associated with it, indicating the preference for this vertex to belong to V_C. Initially, each vertex i is uncoloured (lies in V_U) and is assigned a gain value $gain(i)$ equal to its degree. The gains are held in a priority queue. At each step, a uncoloured vertex with the highest gain is removed from the queue and is moved into V_C. Its neighbours are then moved into V_F. They are also removed from the queue. For each such new vertex in V_F, the gain values of its uncoloured neighbours are increased by one. The procedure is repeated until the queue is empty (that is, until all the vertices belong to either V_C or V_F).

In this algorithm, the gain of an uncoloured vertex is always equal to the number of neighbours in V_U plus twice the number of neighbours in V_F. An uncoloured vertex with a large number of neighbours in V_F is therefore more likely to be moved into V_C. This ensures that V_F vertices are well "covered" by V_C vertices, yielding a more uniform distribution of V_C vertices and a better prolongation operator than the greedy approach. Our maximal independent vertex set algorithm is outlined below.

Maximal independent vertex set algorithm

- *initialisation:*
  - $V_C = \emptyset$
  - $V_U = V$

  - *for each vertex* $i \in V$
    $gain(i) = deg(i)$

- *do while* $V_U \neq \emptyset$

  - $i_{max}: \; gain(i_{max}) = \max_{j \in V_U}(gain(j))$
  - $V_U = V_U \backslash \{i_{max}\}$
  - $V_C = V_C \cup \{i_{max}\}$
  - *for each vertex* $j \in adj(i_{max}) \cap V_U$
    $V_U = V_U \backslash \{j\}$
    for each vertex $k \in adj(j) \cap V_U$
      $gain(k) = gain(k) + 1$

Once the maximal independent vertex set has been computed, the prolongation operator P is defined by (7). The coarse graph vertex weights w_c are calculated as in (8) and the global priority vector p_f^0 for the fine graph is computed as in (9). For `MC60`, the entries of p_f^0 are each rounded to the nearest integer.

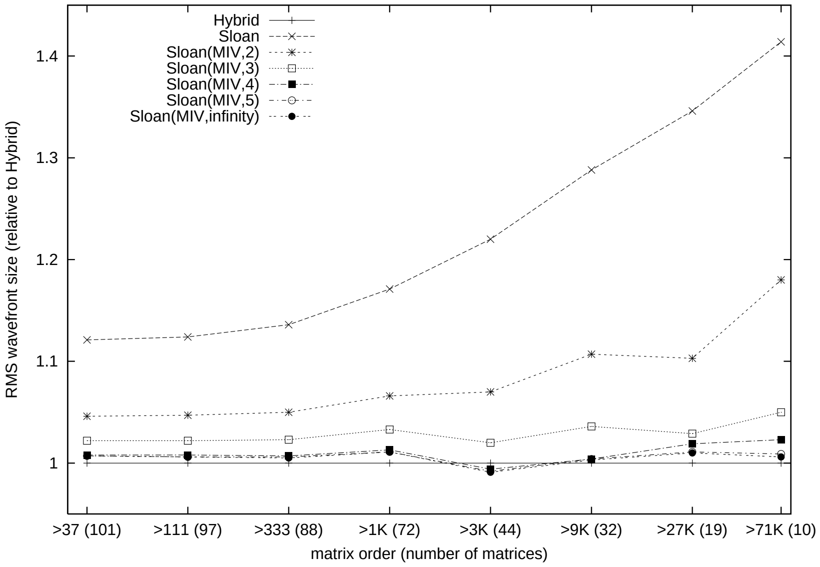

Figure 9: A comparison of the RMS wavefronts for the Sloan, the Hybrid and the $Sloan(MIV, K)$ algorithms.

Figure 9 compares the RMS wavefront for the Sloan and the Hybrid algorithms with the $Sloan(MIV, K)$ algorithm using the above maximal independent vertex set algorithm. It is seen that, as the number of levels increases, the multilevel orderings improve. The multilevel algorithm without a preset maximum number of levels, $Sloan(MIV, \infty)$, produces orderings of comparable quality to the Hybrid algorithm and, in terms of CPU time

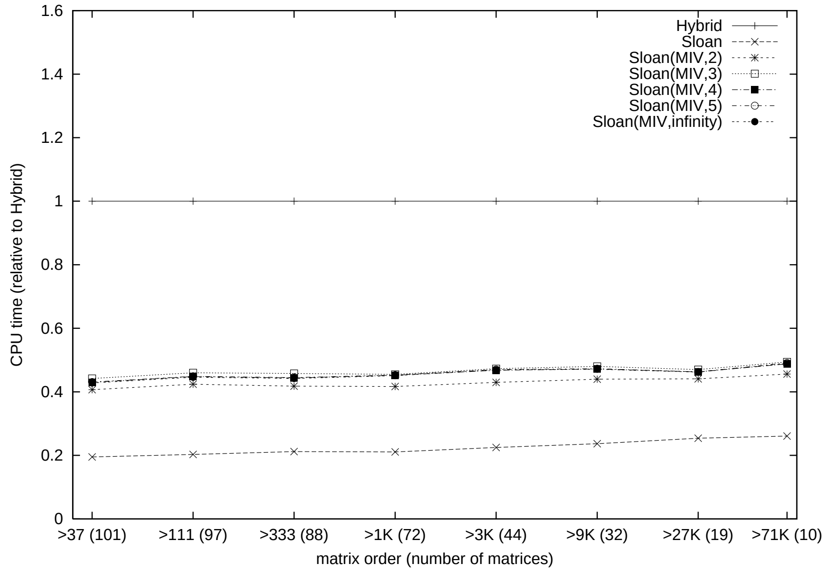

Figure 10: A comparison of the CPU times for the Sloan, the Hybrid and the $Sloan(MIV, K)$ algorithms.

(Figure 10), is substantially faster, requiring about half the time of the Hybrid algorithm. Since $Sloan(MIV, \infty)$ is generally no more expensive in terms of CPU time than $Sloan(MIV, K)$ with $K > 2$ and it produces the smallest RMS wavefronts, we recommend not imposing a maximum number of levels on the multilevel algorithm.

4.3 Sloan versus Hybrid on coarsest graph

The coarsest graph has only a small number of vertices and so it can rapidly be ordered using any appropriate algorithm. In all the results presented so far, the Sloan algorithm has been used but the Hybrid algorithm can be used instead. This gives the multilevel Hybrid algorithm, $Hybrid(MIV, K)$. Figure 11 compares the RMS wavefront for this algorithm with that for the Sloan and the Hybrid algorithms. We see that, for any preset maximum number of levels K, results for the $Hybrid(MIV, K)$ algorithm are comparable to those for the Hybrid algorithm. Even if there are only two levels, the quality of the ordering on the coarsest (level 2) graph is such that the application of a prolongation and refinement step is able to produce a high quality ordering on the fine graph. This is in contrast to $Sloan(MIV, 2)$ where, on the coarsest graph, the Sloan algorithm does not yield such a good ordering. As the number of levels increase, the performance of $Sloan(MIV, K)$ is comparable to $Hybrid(MIV, K)$, indicating that, because the Sloan algorithm performs well on small problems, in terms of quality, the choice between the Sloan and the Hybrid algorithms on the coarsest graph is not important when that graph is small. However, using the Sloan algorithm has the advantage of not requiring

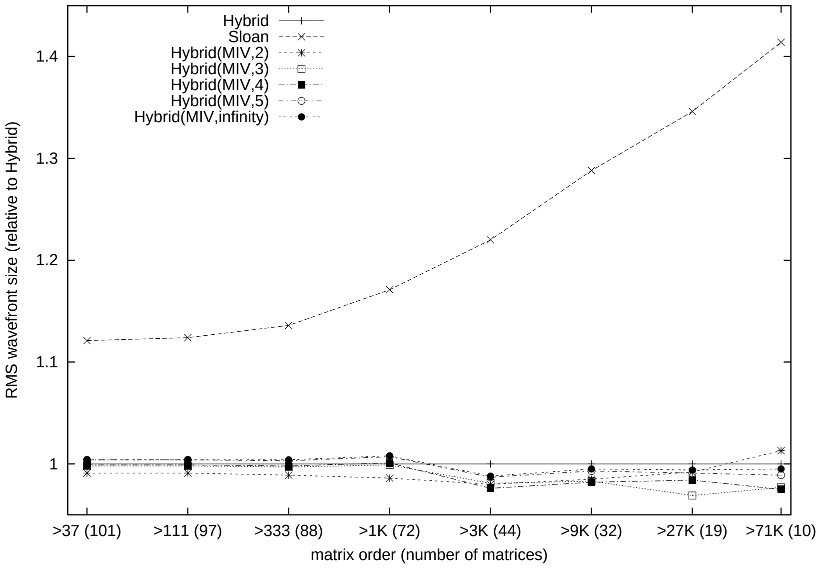

Figure 11: A comparison of the RMS wavefronts for the Sloan, the Hybrid and the $Hybrid(MIV, K)$ algorithms.

any spectral information to be computed.

The fact that the quality of the $Hybrid(MIV, K)$ orderings varies little with the number of levels K indicates that the multilevel process based on the maximal independent set combined with Sloan refinement is of good quality, in the sense that it preserves, if not enhances, the quality of the ordering achieved on the coarsest graph using the Hybrid algorithm. In fact, we could use the quality of the $Hybrid(S, K)$ algorithm to measure the quality of the multilevel scheme based on the coarsening strategy S. For example, Figure 12 shows the RMS wavefront for the $Hybrid(EC, K)$ algorithm. It is seen that in general, the more levels there are, the poorer the quality of the ordering. This is in contrast to Figure 11, and indicates that the edge-collapsing based multilevel approach may not be able to preserve the high quality ordering achieved on the coarsest level.

The CPU times comparisons for the Sloan, the Hybrid and $Hybrid(MIV, K)$ algorithms are given in Figure 13. The $Hybrid(MIV, K)$ algorithm needs only half the CPU time of the Hybrid algorithm. For small K, it is slightly more expensive than $Sloan(MIV, K)$ because of the extra cost associated with using the Hybrid algorithm on the coarsest graph.

Table 1 lists the RMS wavefronts for the Hybrid, $Sloan(MIV, \infty)$ and $Hybrid(MIV, \infty)$ algorithms for each of the test problems of order greater than 10000. The smallest wavefront for each problem (and those within 3 per cent of the smallest) are given in bold. It can be seen that, although the three algorithms on average produce orderings with similar RMS wavefronts, their behaviour on individual matrices can differ significantly. This is typical of heuristic-based algorithms and, for a given problem, which algorithm will

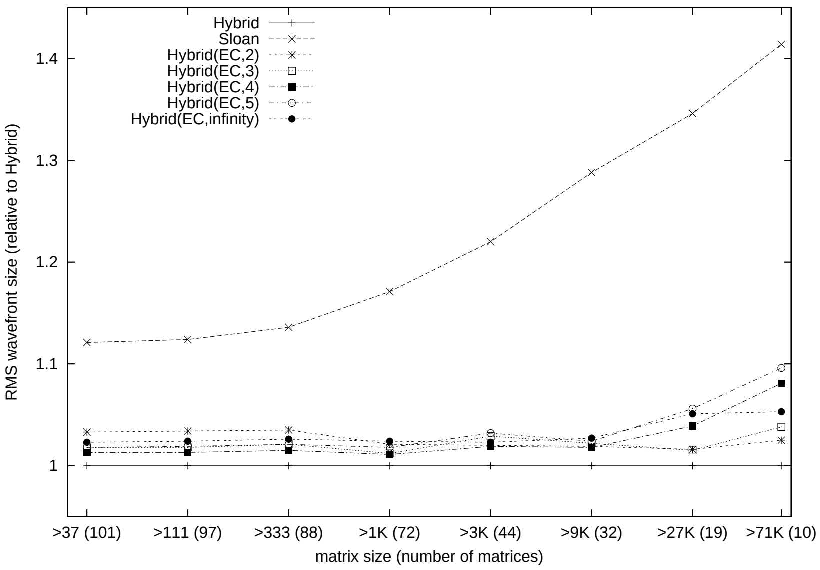

Figure 12: A comparison of the RMS wavefront for the Sloan, the Hybrid and the $Hybrid(EC, K)$ algorithms.

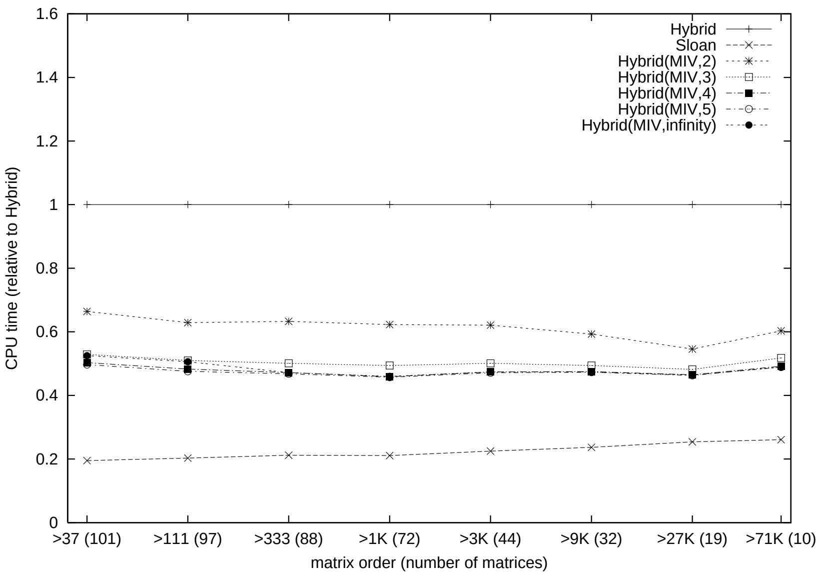

Figure 13: A comparison of the CPU times for the Sloan, the Hybrid and the $Hybrid(MIV, K)$ algorithms.

Table 1: RMS wavefronts for the Hybrid, the $Sloan(MIV, \infty)$ and the $Hybrid(MIV, \infty)$ algorithms on matrices of order > 10000.

Identifier	order	Hybrid	$Sloan(MIV, \infty)$	$Hybrid(MIV, \infty)$
shuttle_eddy	10429	62.55	**60.55**	**59.75**
bcsstk17	10974	**226.63**	**229.63**	240.93
bcsstk18	11948	288.24	**197.25**	**195.37**
bcsstk29	13992	301.50	**192.74**	**192.74**
barth5	15606	**84.20**	97.11	97.75
pds10	16558	**566.94**	680.36	680.36
copter1	17222	401.04	**370.33**	**378.2**
e40r0000	17281	**162.94**	**162.38**	**162.34**
Crplat2	18010	**244.61**	254.16	255.84
tandem_vtx	18454	**288.62**	**287.30**	**282.07**
ford1	18728	**99.42**	108.95	109.56
bcsstk30	28924	**303.03**	**296.99**	321.30
Thread	29736	**1857.44**	1940.28	**1864.85**
bcsstk31	35588	750.81	**526.19**	558.09
finance256	37376	179.18	130.42	**116.95**
bcsstk32	44609	**471.84**	578.90	618.11
skirt	45361	**621.77**	738.62	743.59
nasasrb	54870	**336.84**	344.67	**337.29**
Srb1	54924	**327.74**	**333.39**	**332.48**
copter2	55476	597.79	726.00	**572.89**
finance512	74752	137.16	**114.58**	127.21
onera_dual	85567	**563.14**	697.67	632.99
tandem_dual	94069	451.83	**436.37**	**440.46**
MT1	97578	1035.08	1187.96	**971.93**
ford2	100196	**305.05**	327.58	343.31
Shipsec1	140874	1452.77	1666.05	**1434.77**
Fullb	199187	**1867.09**	1955.66	1992.47
Fcondp2	201822	1713.82	**1542.18**	**1559.93**
Troll	213453	3657.43	**2343.68**	2887.74
Halfb	224617	**1347.94**	1431.09	1486.26

produce the best ordering cannot be predicted *a priori*.

4.4 Sensitivity of the multilevel algorithm

So far, we have established that a multilevel algorithm based on a maximal independent vertex set gives orderings of comparable quality to the Hybrid algorithm, but is significantly faster. This section explores the multilevel algorithm further by studying its sensitivity to the choice of a number of parameters.

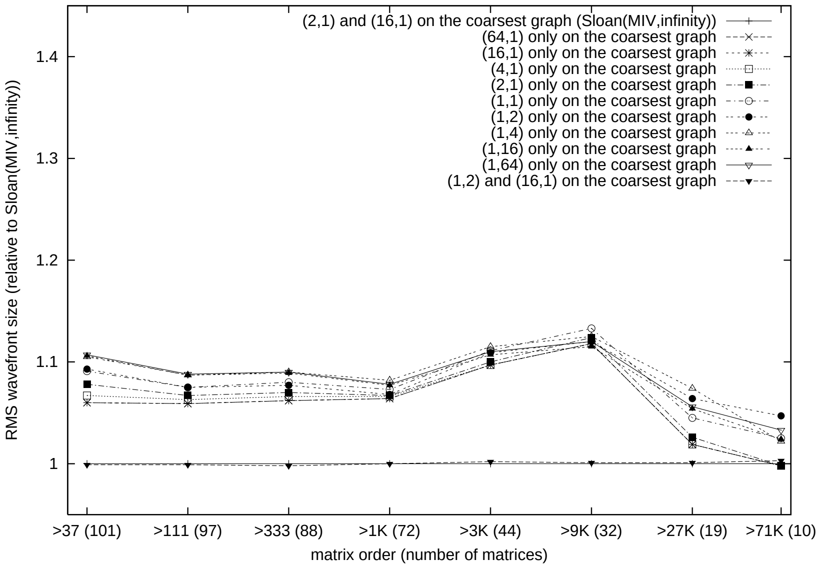

Figure 14: The effect of the coarsest graph ordering on the RMS wavefront of the multilevel algorithm. The $Sloan(MIV, \infty)$ algorithm is used, but with only one weight pair (chosen from $(64, 1)$,. . .,$(1, 64)$), or two weight pairs $(1, 2)$ and $(16, 1)$. All results relative to $Sloan(MIV, \infty)$ with the weight pairs $(2, 1)$ and $(16, 1)$ on the coarsest graph.

We have seen that, with an unlimited number of levels, the coarsest graph ordering based on both the Sloan algorithm ($Sloan(MIV, \infty)$) and the Hybrid algorithm ($Hybrid(MIV, \infty)$) yield orderings of similar quality. Following Reid and Scott [22], in the Sloan algorithm, two orderings are generated from the weight pairs $(2, 1)$ and $(16, 1)$, and the better of the two is chosen. Figure 14 illustrates the effect of using a single pair of weights. If we generate only one ordering based on a single pair of weights (chosen among $(64, 1)$, $(16, 1)$, $(4, 1)$, $(2, 1)$, $(1, 1)$, $(1, 2)$, $(1, 4)$, $(1, 16)$ and $(1, 64)$), the quality of the final ordering obtained using the multilevel algorithm is not as good, although the difference is usually less than 10%. If instead of the two pairs $(2, 1)$ and $(1, 16)$, we use $(1, 2)$ and $(1, 16)$, the difference in the quality of the final ordering is extremely small, indicating that the precise choice for the weights is not critical. As the coarsest graph can be ordered very quickly because of its small size, if it is important to obtain the smallest possible wavefront, it may be worthwhile to try a number of different weights and choose the best ordering among them.

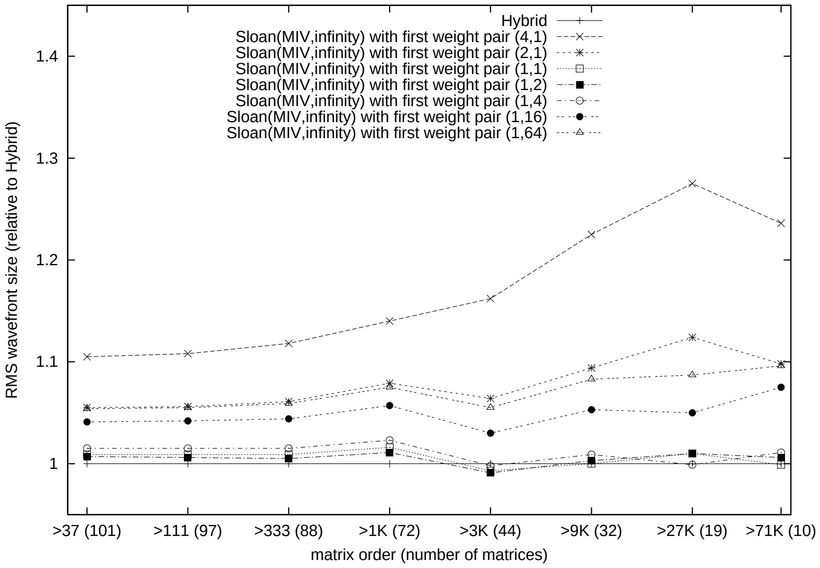

Figure 15: The effect of the first pair of weights during the refinement process on the RMS wavefront of the multilevel algorithm $Sloan(MIV, \infty)$. The second pair of weights is fixed at $(16, 1)$. All results are relative to the Hybrid algorithm.

We have also looked at the sensitivity of the ordering to the choice of weights for the prolongation and refinement stage of the multilevel algorithm. In all the experiments reported so far, we used the weight pairs $(1, 2)$ and $(16, 1)$. This choice of weight pair was recommended in Reid and Scott [22] for the Hybrid algorithm, where it was argued that a larger W_2 in (5) is preferred when $p(i)$ is of good quality. Figure 15 illustrates the effect of varying the first pair of weights on the quality of the ordering given by $Sloan(MIV, \infty)$. In this experiment, the second pair is fixed at $(16, 1)$, while the first pair is allowed to vary between $(4, 1)$ to $(1, 16)$. We see that a weight pair (W_1, W_2) with W_2 slightly larger than or equal to W_1 is beneficial. In general, the weight pairs $(1, 1)$, $(1, 2)$ and $(1, 4)$ all give similar RMS wavefronts. We have also looked at the effect of varying the first pair of weights on the quality of the ordering given by $Hybrid(MIV, \infty)$, and found that the same conclusion can be drawn.

We have stated that the use of vertex weights has a negative effect on the quality of our multilevel algorithm. To illustrate this, Figure 16 shows the RMS wavefront for $Sloan(MIV, \infty)$ and $Sloan(EC, \infty)$, with and without vertex weights. Clearly, for both algorithms the use of vertex weights causes a deterioration in the quality of the final ordering. We should point out that we pass the vertex weights to the code `MC60` in the array `VARS` (in the case of the multilevel algorithm based on the maximal independent set, the vertex weights are first rounded to integers). `MC60` requires `VARS(IS)` to be set to the number of vertices represented by the supervertex `IS`, where supervertices are derived by merging vertices that satisfy (2). Supervertices are therefore not the same as the vertices in the coarse graphs obtained in the multilevel algorithm. For example, vertices in the coarse graph resulting from the edge-collapsing approach are formed by merging vertices that share edges (but not necessarily satisfy (2)). This may explain why we have not been able to use vertex weights in MC60 to advantage.

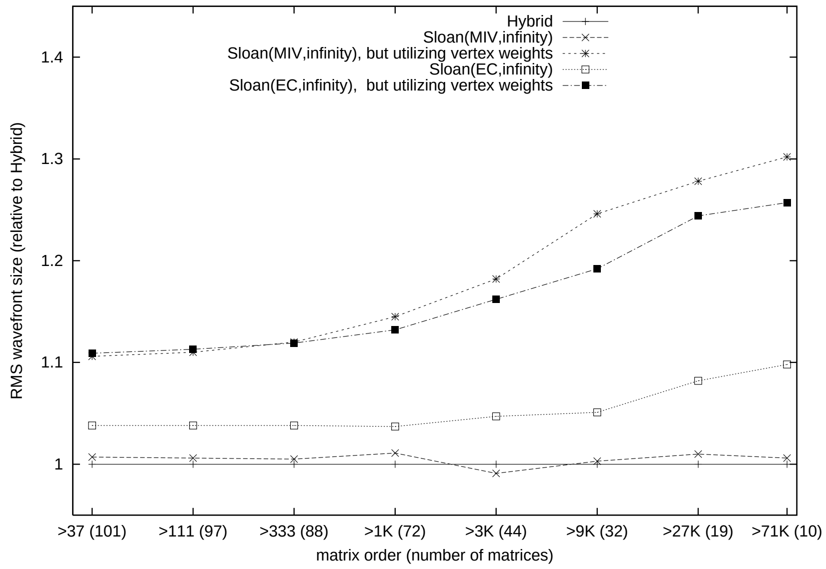

Figure 16: The effect of using vertex weights on the RMS wavefront of the multilevel algorithms $Sloan(MIV, \infty)$ and $Sloan(EC, \infty)$. All results are relative to the Hybrid algorithm.

Finally, as discussed in Section 4.2, the more sophisticated algorithm of Ruge and Stuben [23] for selecting the maximal independent set is favoured over the greedy algorithm. The former tends to yield a more uniform distribution of V_C vertices and a better prolongation operator than the greedy approach. Furthermore, in the Ruge and Stuben algorithm, coarse vertices are selected by maximizing the number of neighbours in V_F and V_U. In general, this gives a more aggressive coarsening, and less dense matrices on the coarse graphs. The result is that a multilevel algorithm based on the Ruge and Stuben approach requires less CPU time than the greedy algorithm. This is illustrated in Figure 17, where $Sloan(MIV, \infty)$ is compared with $Sloan(MIVG, \infty)$, with the latter denoting the multilevel algorithm using the greedy approach for selecting the maximal independent set. $Sloan(MIV, \infty)$ clearly takes less CPU time and yields orderings of slightly higher quality. We have also compared the two approaches for generating the maximal independent set on the $Hybrid(MIV, \infty)$ algorithm, and found that the same conclusion applies.

So far, we have been using the RMS wavefront to compare the algorithms. Results based on the profiles have not been given due to space limitations, although the same conclusions may be drawn if the profile is used as a measure

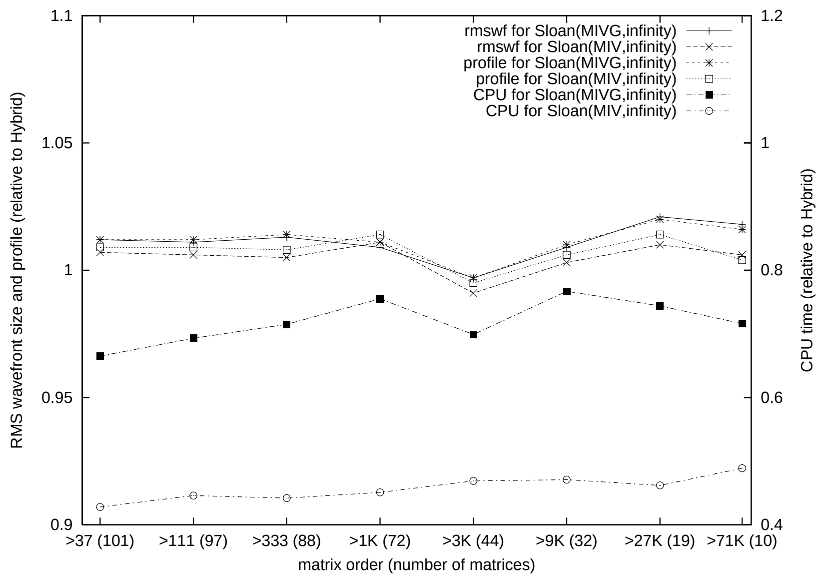

Figure 17: A comparison of the multilevel algorithms based on two approaches of selecting a maximal independent vertex set. $Sloan(MIV, \infty)$ is based on the Ruge and Stuben algorithm [23]. $Sloan(MIVG, \infty)$ is based on the greedy algorithm. All results are relative to the Hybrid algorithm. The left y axis is used for RMS wavefront ($rmsf$) and profile; the right y axis for CPU time.

of ordering quality. Figure 17 includes the profile for the $Sloan(MIV, \infty)$ and $Sloan(MIVG, \infty)$ orderings relative to the Hybrid orderings. As can be seen, the trend for the profile is very similar to that of the RMS wavefront.

5 Conclusions and Future Work

New multilevel reordering algorithms for minimising the profile and wavefront of symmetric matrices have been considered in this paper. A number of approaches have been investigated. A multilevel algorithm, combining a coarsening strategy based on the maximal independent set with the Sloan or Hybrid algorithms on the coarsest graph, has been found to give orderings of similar quality to that of the best existing algorithm (the Hybrid algorithm of Kumfert and Pothen [17]), whilst being significantly faster, requiring on average only half the CPU time. Of particular note is the multilevel Sloan algorithm. With no limit imposed on the maximum number of levels, this algorithm has been shown to yield orderings of similar quality to that of the Hybrid algorithm, without requiring any spectral information.

We are investigating the possibility of further improving the multilevel algorithm so that it consistently outperforms the Hybrid algorithm, both in terms of CPU time and ordering quality. We believe that to achieve this goal it will be necessary to utilise the vertex and edge weights of the coarse graphs. We are looking at the best mechanism of including this information in the reordering and refinement of the coarse graphs. Another possible way of improving the multilevel algorithm is to use a more sophisticated ordering algorithm on the coarsest graph and then to look at translating improvements in the quality of the ordering on the coarsest graph into corresponding improvements on the original fine graph.

It may also be possible to extend our multilevel approach to the ordering of unsymmetric matrices for use with frontal solvers. This will build on the work of Scott [24] on row ordering algorithms and the work of Hu, Maguire, and Blake [12] on applying a multilevel algorithm for reordering unsymmetric matrices into bordered form.

6 Acknowledgements

We would like to thank Iain Duff and John Reid for helpful comments on a draft of this paper. We are also grateful to Christian Damhaug and Gary Kumfert for supplying some of the test problems used in this paper.

References

[1] S. T. Barnard, A. Pothen, and H. D. Simon. A spectral algorithm for envelope reduction of sparse matrices. *Numerical Linear Algebra with Applications*, 2:317–334, 1995.

[2] S. T. Barnard and H. D. Simon. Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Concurrency: Practice and Experience*, 6:101–117, 1994.

[3] E. G. Bolman and B. Hendrickson. A multilevel algorithm for reducing the envelope of sparse matrices. Technical report 96-21, Stanford University, 1996.

[4] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proc. 24th Nat. Conf. ACM*, pages 157–172. ACM Publications, 1969.

[5] I. S. Duff, J. K. Reid, and J. A. Scott. The use of profile reduction algorithms with a frontal code. *International Journal for Numerical Methods in Engineering*, 28:2555–2568, 1989.

[6] A. George. *Computer implementation of the finite-element method*. PhD thesis, Department of Computer Science, Stanford Univeristy, 1971.

[7] N. Gibbs, W. Poole, and P. Stockmeyer. An algorithm for reducing the bandwidth and profile of a sparse matrix. *SIAM Journal of Numerical Analysis*, 13:236–250, 1976.

[8] N. E. Gibbs. Algorithm 509: a hybrid profile reduction algorithm. *ACM Transactions on Mathematical Software*, 2:378–387, 1976.

[9] B. Hendrickson and R. Leland. The Chaco User's Guide, version 1.0. Technical report sand93-2339, Sandia National Laboratories, Allbuquerque, NM, 1993.

[10] B. Hendrickson and R. Leland. A multilevel algorithm for partitioning graphs. Technical report sand93-1301, Sandia National Laboratories, Allbuquerque, NM, 1993.

[11] HSL. A collection of Fortran codes for large scale scientific computation, 2000. Full details from `http://www.numerical.rl.ac.uk/hsl`.

[12] Y. F. Hu, K. C. F. Maguire, and R. J. Blake. A multilevel unsymmetric matrix ordering algorithm for parallel process simulation. *Computers and Chemical Engineering*, 23:1631–1647, 2000.

[13] G. Karypis and V. Kumar. Parallel multilevel graph partitioning. Technical report, Department of Computer Science, University of Minnesota, Minneapolis, MN 55455, 1995.

[14] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48:96–129, 1998.

[15] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20:359–392, 1999.

[16] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell Systems Tech. J.*, 49:291–308, 1970.

[17] G. Kumfert and A. Pothen. Two improved algorithms for envelope and wavefront reduction. *BIT*, 35:1–32, 1997.

[18] Y. Lin and J. Yuan. Minimum profile of grid networks in structure analysis. Preprint, Department of Methematics, Zhengzhou University, Zhengzhou, Henan 450052, People's Republic of China, 1993.

[19] J. W. H. Liu and A. H. Sherman. Comparative analysis of the Cuthill-McKee and reverse Cuthill-McKee ordering algorithms for sparse matrices. *SIAM J. Numer. Anal.*, 13:198–213, 1976.

[20] G. H. Paulino, I. F. M. Menezes, M. Gattass, and S. Mukherjee. Node and element resequencing using the Laplacian of a finite element graph, Part I. *International Journal for Numerical Methods in Engineering*, 37:1511–1530, 1994.

[21] G. H. Paulino, I. F. M. Menezes, M. Gattass, and S. Mukherjee. Node and element resequencing using the Laplacian of a finite element graph, Part II. *International Journal for Numerical Methods in Engineering*, 37:1531–1555, 1994.

[22] J. K. Reid and J. A. Scott. Ordering symmetric sparse matrices for small profile and wavefront. *International Journal for Numerical Methods in Engineering*, 45:1737–1755, 1999.

[23] J. W. Ruge and K. Stüben. Algebraic multigrid (AMG). In S. F. McCormick, editor, *Multigrid Methods*, volume 3 of *Frontiers in Applied Mathematics*, pages 73–130. SIAM, Philadelphia, PA, 1987.

[24] J. A. Scott. A new row ordering strategy for frontal solvers. *Numerical Linear Algebra with Applications*, 6:1–23, 1999.

[25] H. D. Simon. Partitioning of unstructured problems for parallel processing. *Computer Systems in Engineering*, 2:135–148, 1991.

[26] S. W. Sloan. An algorithm for profile and wavefront reduction of sparse matrices. *International Journal for Numerical Methods in Engineering*, 23:239–251, 1986.

[27] C. Walshaw, M. Cross, and M. Everett. Dynamic mesh partitioning: a unified optimisation and load-balancing algorithm. Technical report 95/im/06, University of Greenwich, London SE18 6PF, UK, 1995.

A The test problems

Table 2: The suite of test problems. $rmsf$ is the initial RMS wavefront and ρ is the ratio between the RMS wavefronts before and after reordering with the Hybrid algorithm.

Identifier	$\|V\|$	$\|E\|$	$rmsf$	ρ
1138_bus	1138	1458	87.00	6.88
barth	6691	19748	2673.11	42.86
barth4	6019	17473	404.62	7.56
barth5	15606	45878	284.36	3.38
Baug	9600	232980	1459.93	6.07
bcspwr06	1454	1923	57.68	4.92
bcspwr07	1612	2106	61.00	4.99
bcspwr08	1624	2213	64.35	5.46
bcspwr09	1723	2394	308.51	21.86
bcspwr10	5300	8271	1294.66	47.68
bcsstk08	1074	5943	239.84	1.80
bcsstk11	1473	16384	104.34	2.18
bcsstk12	1473	16384	104.34	2.18
bcsstk13	2003	40940	229.18	0.97
bcsstk14	1806	30824	115.23	1.22
bcsstk15	3948	56934	263.35	1.44
bcsstk17	10974	208838	261.87	1.16
bcsstk18	11948	68571	468.72	1.63
bcsstk21	3600	11500	119.39	2.17
bcsstk23	3134	21022	353.47	1.50
bcsstk24	3562	78174	613.47	4.98
bcsstk28	4410	107307	190.39	1.42
bcsstk29	13992	302748	551.89	1.83
bcsstk30	28924	1007284	641.77	2.12
bcsstk31	35588	572914	672.80	0.90
bcsstk32	44609	985046	2905.61	6.16
bcsstm12	1473	9093	103.80	3.79
bcsstm13	2003	9970	52.36	0.87
blckhole	2132	6370	93.98	1.67
can_1054	1054	5571	274.78	8.83

Table 3: The suite of test problems (continued).

Identifier	$\|V\|$	$\|E\|$	$rmsf$	ρ
can_1072	1072	5686	279.31	8.41
commanche_dual	7920	11880	2397.83	55.98
copter1	17222	96921	1127.23	2.81
copter2	55476	352238	21892.02	36.62
Crplat2	18010	471468	1286.38	5.26
dwg961b	961	4815	179.24	7.10
dwt	607	2262	55.43	2.08
dwt1005	1005	3808	137.66	4.03
dwt1007	1007	3784	26.93	1.31
dwt1242	1242	4592	105.20	3.18
dwt162	162	5100	18.95	2.02
dwt193	193	1650	43.84	1.80
dwt198	198	5970	30.90	4.37
dwt209	209	7670	50.32	3.44
dwt221	221	7040	50.39	5.54
dwt234	234	3000	9.36	1.48
dwt245	245	6080	18.48	1.82
dwt2680	2680	11173	234.42	6.90
dwt307	307	1108	27.36	1.06
dwt310	310	1069	9.85	1.02
dwt346	346	1440	27.15	1.36
dwt361	361	1296	15.38	1.09
dwt419	419	1572	107.07	5.50
dwt492	492	1332	79.51	8.92
dwt503	503	2762	78.60	2.77
dwt512	512	1495	14.55	1.24
dwt59	59	1040	8.22	1.72
dwt592	592	2256	55.18	2.96
dwt607	607	2262	55.43	2.08
dwt66	66	1270	11.01	3.74
dwt72	72	7500	3.46	1.03
dwt758	758	2618	37.95	3.65
dwt869	869	3208	25.02	1.49
dwt87	87	2270	29.38	4.68
dwt878	878	3285	31.92	1.38
dwt918	918	3233	131.14	6.58

Table 4: The suite of test problems (continued).

Identifier	$\|V\|$	$\|E\|$	$rmsf$	ρ
dwt992	992	7876	301.99	8.86
e40r0000	17281	270737	438.20	2.69
eris1176	1176	8688	81.59	3.59
Fcondp2	201822	5546247	10322.91	6.02
finance256	37376	130560	7441.93	41.53
finance512	74752	261120	14831.49	108.13
ford1	18728	41424	1954.25	19.66
ford2	100196	222246	4282.70	14.04
Fullb	199187	5754445	45506.16	24.37
Halfb	224617	6081602	35656.53	26.45
jagmesh4	1440	4032	32.62	1.66
jagmesh5	1180	3285	32.57	1.69
jagmesh7	1138	3156	39.52	2.13
jagmesh8	1141	3162	32.17	1.38
jagmesh9	1349	3876	54.67	2.31
lshp3466	3466	10215	109.46	2.38
mhd4800b	4800	11360	17.11	4.19
MT1	97578	4827996	2815.98	2.72
nasasrb	54870	1311227	401.75	1.19
nos7	729	1944	76.26	1.16
onera_dual	85567	166817	9336.32	16.58
pds10	16558	66550	1129.78	1.99
plat1919	1919	15240	739.36	16.38
qc2534	2534	230413	177.61	1.00
s3rmt3m3	5357	101169	478.58	3.81
Shipsec1	140874	3836265	3290.65	2.27
shuttle_eddy	10429	46585	1161.54	18.57
skirt	45361	1268228	1092.01	1.76
Srb1	54924	1453614	1527.03	4.66
sstmodel	3345	9702	34.27	0.61
tandem_dual	94069	183212	5831.87	12.91
tandem_vtx	18454	117448	4705.44	16.30
Thread	29736	2220156	6676.39	3.59
Troll	213453	5885829	4703.06	1.29
zenios	2873	12159	431.21	54.54