NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. 2014; **21**:51–67
Published online 11 October 2012 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/nla.1859

Level-based heuristics and hill climbing for the antibandwidth maximization problem

Jennifer Scott [1,*,†] and Yifan Hu [2]

[1] *Computational Science and Engineering Department, Rutherford Appleton Laboratory, Chilton, Oxfordshire, OX11 0QX, UK*
[2] *AT&T Labs Research, 180 Park Avenue, Florham Park, NJ 07932, USA*

SUMMARY

The antibandwidth maximization problem aims to maximize the minimum distance of entries of a sparse symmetric matrix from the diagonal and as such may be regarded as the dual of the well-known bandwidth minimization problem. In this paper, we consider the feasibility of adapting heuristic algorithms for the bandwidth minimization problem to the antibandwidth maximization problem. In particular, using an inexpensive level-based heuristic, we obtain an initial ordering that we refine using a hill-climbing algorithm. This approach performs well on matrices coming from a range of practical problems with an underlying mesh. Comparisons with existing approaches show that, on this class of problems, our algorithm can be competitive with recently reported results in terms of quality while being significantly faster and applicable to much larger problems. Copyright © 2012 John Wiley & Sons, Ltd.

Received 27 October 2011; Revised 22 May 2012; Accepted 9 September 2012

KEY WORDS: antibandwidth maximization; sparse matrices; level sets; hill climbing

1. BACKGROUND AND MOTIVATION

Since the 1960s, considerable attention has been paid to the design and development of algorithms for minimizing the bandwidth of a sparse symmetric matrix $A = \{a_{ij}\}$, that is, finding a labelling (or ordering) of the rows and columns of A that minimizes the maximum distance b from the diagonal

$$b = \min_i \left\{ \max_j \{|i - j| : a_{ij} \neq 0\} \right\}$$

(see, for example, [1–5]). Until relatively recently, much less attention has focused on the antibandwidth maximization problem, which is the problem of finding a labelling of the rows and columns of A that maximizes the minimum distance ab from the diagonal

$$ab = \max_i \left\{ \min_j \{|i - j| : i \neq j \text{ and } a_{ij} \neq 0\} \right\}.$$

Many algorithms for reducing the bandwidth of A make extensive use of the adjacency graph $\mathcal{G}$ of A. This is an undirected graph that has a node for each row (or column) of the matrix, and node i is a neighbour of node j if a_{ij} (and by symmetry a_{ji}) is an entry (nonzero) of A. In terms

*Correspondence to: Jennifer Scott, Computational Science and Engineering Department, Rutherford Appleton Laboratory, Chilton, Oxfordshire, OX11 0QX, UK.
†E-mail: jennifer.scott@stfc.ac.uk

of graphs, the antibandwidth maximization problem is to label the nodes of the graph such that the length of the shortest edge is maximized (that is, the labelling difference of the end nodes among all edges is maximized). This problem was introduced by Leung *et al.* [6] in 1984 in connection with multiprocessor scheduling problems. It is also referred to as the *dual bandwidth* problem [7] or the *separation* problem [6]. It arises in a number of practical applications. For example, it belongs to the family of obnoxious facility location problems [8]. Here, the 'enemy' graph is represented by n people, and there is an edge between two people iff they are enemies. The problem is to build each person a house along a road so that the minimal distance between enemies is maximized. Another example is the radio frequency assignment problem in which the nodes correspond to transmitters, and the edges are between interfering transmitters; the objective is to assign the frequencies so that those for the interfering transmitters are as different as possible.

Like the bandwidth minimization problem, the antibandwidth maximization problem is NP-complete [6]. In the literature, theoretical results have been presented for some special graphs, including paths, cycles, rectangular grids, special trees and complete bipartite graphs (see, for example, [9] and the references therein). Recently, there has been an interest in developing algorithms to compute solutions that are close to the optimal and that apply to general graphs. In particular, Hu, Kobourov and Veeramoni [10] have developed an algorithm `GSpectral` (Greedy Spectral) that is based on computing the eigenvector corresponding to the largest eigenvalue of the Laplacian associated with the graph and then using a greedy refinement algorithm. They have applied this to the maximum differential graph colouring problem and reported results for some small examples. Duarte, Martí, Resende and Silva [11] have proposed an integer linear programming formulation and several heuristics based on Greedy Randomized Adaptive Search Procedure (`GRASP`) with path relinking. They present some high-quality computational results for general graphs, although the run times for their relatively modest-sized test problems (graphs with fewer than 9000 nodes) are quite high (typically several minutes for their fastest approach applied to their largest problems). Thus, we would like to develop alternative algorithms for increasing the antibandwidth that are significantly faster while retaining good quality. This paper is the first step in achieving this aim.

An important and well-known example of a bandwidth reduction algorithm is the Cuthill–McKee algorithm [1] and its many variants, including the Gibbs–Poole–Stockmeyer algorithm [2]. The Cuthill–McKee algorithm constructs a level-set structure of the graph $\mathcal{G}$ and labels the nodes according to these levels. In this paper, we consider the feasibility of modifying this approach to obtain a practical algorithm for increasing the antibandwidth of A. We find that on its own, this is not generally sufficient to yield large antibandwidths but that when combined with a suitable refinement algorithm, we are able to compute high-quality orderings for problems that arise from a range of applications with an underlying mesh. Furthermore, our approach is fast and thus potentially practical for larger problems that cannot be tackled by either the `GSpectral` or `GRASP` approaches.

The rest of this paper is organized as follows. We begin (Section 2) by briefly recalling the Cuthill–McKee algorithm and then considering how it might be modified for the antibandwidth maximization problem. In Section 3, we look at modifying the hill-climbing algorithm of Lim, Rodrigues and Xiao [3] to improve a given ordering. In Section 4, our proposed algorithms are used to reorder a set of test matrices, and our results are compared with those of for `GSPectral` and `GRASP` in Section 5. We summarize our findings and discuss future work in Section 6.

2. LEVEL-BASED APPROACH TO ANTIBANDWIDTH PROBLEM

In this section, we first recall the Cuthill–McKee algorithm for bandwidth minimization and then consider how it may be adapted for the antibandwidth maximization problem.

2.1. The Cuthill–McKee algorithm

Given a starting node s, the Cuthill–McKee algorithm proceeds by relabelling the nodes of the adjacency graph $\mathcal{G}$ by order of increasing distance from s. The algorithm is outlined in Figure 1.

```
Algorithm 1: Cuthill-McKee
Label $s$ as node 1; $l_1 = \{s\}$; $i = 1$
do $k = 2, 3, \ldots$ until $i = n$
    $l_k = \{\}$
    do for each $v \in l_{k-1}$ in label order
        do for each neighbour $u$ of $v$ that has not been labelled,
                in order of increasing degree
            add $u$ to $l_k$; $i = i + 1$; label $u$ as node $i$
        end do
    end do
end do
```

Figure 1. Cuthill–McKee ordering algorithm.

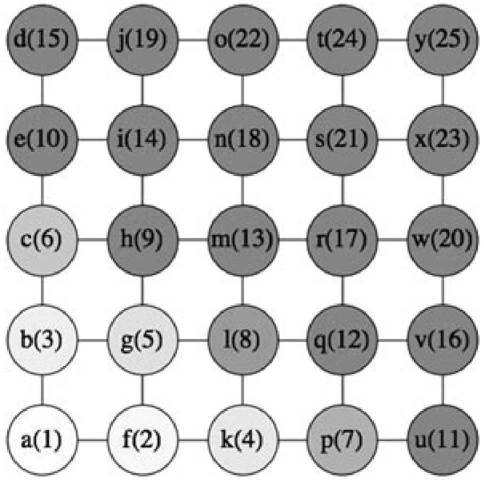

Figure 2. Cuthill–McKee labelling of 5×5 mesh. Here, the letters indicate the initial ordering, and the numbers in parentheses are the new labels. A grey scale colouring is used, with the first node given the lightest shade and the last node the darkest shade.

Here, the degree of a node i is defined as the number of neighbours it has (that is, the number of nodes $j \neq i$ for which $a_{ij} \neq 0$). If $\mathcal{G}$ has more than one component, the procedure is repeated from a starting node in each component.

Ordering the nodes in this way groups them into *level sets*, that is, nodes at the same distance from the starting node. Because nodes in level set l_k can have neighbours only in level sets l_{k-1}, l_k and l_{k+1}, the reordered matrix is block tridiagonal with blocks corresponding to the level sets. It is therefore desirable that the level sets be small, which is likely if there are many of them. The level-set structure rooted at s is denoted by $L(s) = \{l_1, l_2, \ldots, l_h\}$. Algorithms for finding a good starting node are usually based on finding a pseudodiameter of $\mathcal{G}$ (a pair of nodes that are a maximum distance apart or nearly so). Much effort has gone into efficiently finding a pseudodiameter; algorithms are generally based on either using level sets (see, for example, [2] and [12] and the references therein) or using the Fiedler vector [13]. The (non-unique) optimal bandwidth minimization ordering that is obtained by applying the Cuthill–McKee algorithm to a 5×5 mesh is illustrated in Figure 2. Here, the letters indicate the initial ordering, and the numbers in parentheses are the new labels. A grey scale colouring is used, with the first node given the lightest shade and the last node the darkest shade. Node ‘a’ is chosen as the starting node (but any of the other corner nodes could equally well have been chosen).

2.2. *Level-based approach for antibandwidth maximization problem*

In their paper, Miller and Pritikin [14] establish tight bounds for the maximum antibandwidth for various classes of graphs. Further bounds were recently presented by Raspaud *et al.* [9]. In particular, Raspaud *et al.* showed that for a two-dimensional $m \times k$ mesh with $m \geqslant k \geqslant 2$, the lower bound on the maximum antibandwidth proved by Miller and Pritikin is precise; that is, the maximum antibandwidth for such problems is

$$ab = \left\lceil \frac{k(m-1)}{2} \right\rceil. \tag{2.1}$$

Miller and Pritikin described how this bound can be achieved. Nodes i and j with coordinates (p,q) and (p',q') in the mesh are neighbours if and only if $p = p'$ and $|q - q'| = 1$ or $q = q'$ and $|p - p'| = 1$. Miller and Pritikin set the origin $(0,0)$ to a corner of the mesh and define $X = \{(p,q) : p + q \text{ is odd}\}$ and $Y = \{(p,q) : p + q \text{ is even}\}$. X is ordered lexicographically, and then Y is ordered lexicographically. This is equivalent to choosing the starting node s to be a corner of the mesh, constructing the level-set structure $L(s)$, and then taking the level sets in the order $\{l_2, l_4, \ldots, l_h, l_1, l_3, \ldots, l_{h-1}\}$ (here, h is assumed to be even) and ordering nodes in each level set in turn, in natural order.

For a three-dimensional $m \times m \times m$ mesh, Török and Vrt'o [15] show that

$$ab = \frac{4m^3 - 3m^2}{8} + \mathcal{O}(m). \tag{2.2}$$

Their algorithm for labelling the nodes to achieve this optimal value (up to the third order term) again labels the even numbered level sets and then the odd numbered level sets, starting from a corner.

Choosing to start at a corner of the mesh is equivalent to selecting the starting node to be an end point of a diameter of $\mathcal{G}$. This suggests that for more general problems, we should select s to be an end point of a pseudodiameter, construct $L(s)$ and use the level sets to guide the relabelling. In the Cuthill–McKee algorithm, at each stage the list of candidate nodes for the next label comprises the unlabelled neighbours of the nodes that have already been labelled. Thus, a node and its neighbours receive labels that are close to each other, yielding a narrow bandwidth. When attempting to increase the antibandwidth, we need to do the opposite; that is, if a node has been labelled, avoid labelling its neighbours for as long as possible. The approach of Miller and Pritikin does exactly that for mesh problems because none of the nodes in each of the level sets l_r has neighbours in the same level set (the neighbours all belong to the level sets l_{r-1} and l_{r+1}). For more general problems, we will have neighbours belonging to the same level set, and so we need to use a strategy to avoid labelling these neighbours too soon. Our algorithm is outlined in Figure 3.

```
Algorithm 2: Level-based antibandwidth ordering (LB)
Input: starting node $s$, rooted level-set structure $L(s) = \{l_1, l_2, ..., l_h\}$.
Initialise: $sweep = 0$; $flag(1:n) = 0$; $i = 0$.
do until $i = n$
    $sweep = sweep + 1$
    do $r = 1, ..., h$
        do for each unlabelled $u \in l_r$
            if $(flag(u) = sweep)$ cycle
            $i = i + 1$; give node $u$ label $i$
            Set $flag(v) = sweep$ for each unlabelled neighbour $v$ of $u$
        end do
    end do
end do
```

Figure 3. Level-based antibandwidth ordering algorithm.

Algorithm 2 has a number of sweeps, or passes, through the level-set structure. On each sweep, a flag is used to indicate whether or not a node that has yet to be labelled is a candidate for labelling during that sweep. Initially, all flags are set to zero. When a node u is labelled, all the neighbours v of u that are unlabelled are flagged with the current sweep number; these nodes are not candidates for labelling until the next sweep. Note that nodes in level set l_r can only have neighbours in l_{r-1}, l_r and l_{r+1}. For the mesh problems of Miller and Pritikin, Algorithm 2 reduces to ordering the even-numbered level sets and then the odd-numbered level sets. The run time complexity of Algorithm 2 is $\mathcal{O}(d_{\max} * nz)$, where $d_{\max}$ is the maximum degree of a node and nz is the number of edges.

The use of Algorithm 2 to order a 5×5 mesh is illustrated in Figure 4. Again, the letters indicate the initial ordering, and the numbers in parentheses are the new labels; node 'a' is chosen as the starting node.

The success of Algorithm 2 depends on the choice of the starting node s and the sizes of the level sets. The rationale for choosing s to be an endpoint of a pseudodiameter is that it will tend to lead to a long thin level-set structure that is hopefully also well balanced in the sense that (with the exception of the first and last few levels) the levels each have a similar number of entries. We want to avoid $L(s)$ having one level set (or a small number of level sets) l_r that is larger than the other level sets. For suppose a node $u \in l_r$ has a large number of neighbours, most Of which also belong to l_r. Once u has been labelled, if there remains no unlabelled nodes in the other level sets, the remaining unlabelled nodes in l_r will be labelled consecutively. This results in an antibandwidth of 1, even though with the new labelling

$$\min_j \{|i - j| : i \neq j \text{ and } a_{ij} \neq 0\}$$

is significantly larger than 1 for all $i \neq n - k$ for small k. This is illustrated in Section 4.1. To help assess the quality of our labelling, we define the *average antibandwidth* as

$$av = \frac{1}{n} \sum_i \min_j \{|i - j| : i \neq j \text{ and } a_{ij} \neq 0\}. \tag{2.3}$$

Note that this definition considers upper and lower triangular entries, and thus, maximizing av is **not** analogous to the problem of minimizing the profile of A. Recall that the profile is defined to be

$$\textit{profile} = \frac{1}{n} \sum_i \max_j \{i - j : j < i \text{ and } a_{ij} \neq 0\} \tag{2.4}$$

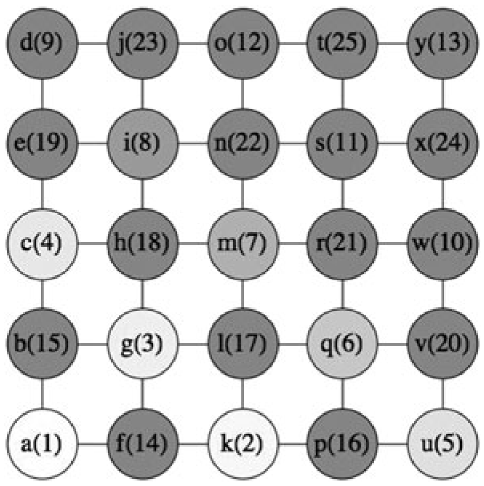

Figure 4. LB labelling of 5×5 mesh. Here, the letters indicate the initial ordering, and the numbers in parentheses are the new labels. A grey scale colouring is used, with the first node given the lightest shade and the last node the darkest shade.

and thus involves minimizing the sum of the maximum distance from the diagonal in the lower triangular part of A only.

3. HILL-CLIMBING REFINEMENT ALGORITHM

The level-based algorithm is a global algorithm. In this section, we look at how we might locally refine the ordering to improve the antibandwdith. Lim *et al.* [3] propose a hill-climbing algorithm for reducing the bandwidth of a symmetric matrix. This local search strategy was adapted for unsymmetric matrices by Reid and Scott [16]. Here, we propose using hill climbing to increase the antibandwidth.

The idea behind hill climbing is that at each step a search is made for a non-critical node to swap with a critical node. For the antibandwidth maximization problem, i is defined to be *critical* if

$$\min_k \{|i-k| : i \neq k \text{ and } a_{ik} \neq 0\} = ab. \tag{3.1}$$

If i is critical, we look for a non-critical j such that symmetrically permuting i and j (that is, swapping rows i and j and columns i and j) leaves both i and j non-critical. Because for increasing the antibandwidth we need to move entries away from the diagonal, candidates for swapping must be sufficiently far apart. In particular, if i is critical and there is some $k < i$ such that $i - k = ab$, then if j lies in the range

$$i - 2 * ab \leqslant j \leqslant i - 1,$$

swapping i and j will not lead to an increase in the antibandwidth. Thus, j is only a swap candidate if it lies outside this range. Similarly, if i is critical and for some $k > i$, $k - i = ab$, to be a swap candidate j must lie outside the range

$$i + 1 \leqslant j \leqslant i + 2 * ab.$$

If j is a candidate for swapping with i, it is necessary to check the entries in both rows i and j to see if a swap is possible. A swap is not acceptable if one or more of the following holds:

1. $a_{ij} \neq 0$ and $|i - j| = ab$.
2. There exists l such that $a_{il} \neq 0$ and $|l - j| \leqslant ab$.
3. There exists k such that $a_{kj} \neq 0$ and $|k - i| \leqslant ab$.

If one of these holds, swapping i and j either decreases the antibandwidth or does not reduce the number of critical nodes. Each accepted swap while the antibandwidth is *ab* reduces the number of critical nodes by one. If the number of critical nodes becomes zero, we restart with antibandwidth $ab + 1$ and repeat the process until no further swaps can be made to reduce the number of critical nodes. The algorithm is summarized in Figure 5. Note that hill climbing cannot decrease the antibandwidth but may decrease the average antibandwidth.

```
Algorithm 3: Hill climbing (HC)
outer: do
    Form the set $V_c$ of critical nodes
    do until $V_c$ is empty
        if (there are nodes $i \in V_c$ and $j \notin V_c$ such that
            swapping $i$ and $j$ leaves both non-critical) then
            swap $i$ and $j$ and remove $i$ from $V_c$
        else
            exit outer
        end if
    end do
end do outer
```

Figure 5. Hill-climbing algorithm.

The differences between hill climbing for reducing the bandwidth and for increasing the antibandwidth are (a) the definition of a critical node and (b) the checks that are needed for finding a suitable swap. For bandwidth reduction, it is sufficient to keep track of the first and last entries in each row. For increasing the antibandwidth, the checking is more expensive because we must check each of the entries in rows i and j (unless the entries are in order of increasing column index, but maintaining this ordering after a swap is also expensive). In our implementation, if i is critical, we swap i with the first suitable j that we find: there is no attempt to find the 'best' j (that is, the j that maximizes $\min_k\{|i-k| : i \neq k \text{ and } a_{jk} \neq 0\}$ and $\min_k\{|j-k| : j \neq k \text{ and } a_{ik} \neq 0\}$). This is partly because of the additional cost that locating the best j at each stage incurs but also because finding the best j does not necessarily lead to the best final antibandwidth. When looking for a swap, we search the rows in reverse order because we found that this generally yielded better results.

4. NUMERICAL EXPERIMENTS

We start by introducing our test problems. Our first set consists of the 24 two-dimensional meshes that are used by Duarte *et al.* [11]. They are constructed as the Cartesian product of two paths, and optimal solutions for the antibandwidth maximization problem are known by construction (see [9]). Our second set (Table I) is taken from the University of Florida Sparse Matrix Collection [17] and comes from the DNVS, HB, INFRO, Schenk-AFE and AG-Monien groups. The order n ranges from 54 to 504 855. For problems with an unsymmetric sparsity structure, we work with $A + A^T$. The algorithms we propose are primarily designed for problems with underlying meshes, and this has influenced our choice of test examples, although we emphasize that most of these do not have a regular rectangular grid structure (see, for example, Figure 6). Some of the smaller problems were chosen because they appear in the paper by Duarte *et al.* [11].

The implementations of our algorithms used in this paper are written in Fortran 95; all experiments are performed on a single core of an Intel Xeon E5620 using the gfortran compiler (version 4.4.3) with the -O3 option.

Table I. Test problems.

Problem	n	nz	Description
HB/curtis54*	54	291	Stiff ordinary differential equation
HB/dwt_234	234	834	2D structural engineering problem
HB/saylr1	238	1128	14×17 grid
AG-Monien/grid1	252	952	2D finite-element problem
HB/nos7	330	4617	$9 \times 9 \times 9$ grid
HB/can_445*	445	3809	Finite-element mesh from aircraft design
HB/nos5	468	5172	3D finite-element approximation of building
HB/662_bus*	662	2474	Model of power system network
HB/nos6*	675	3255	Poisson's equation in L-shaped region
HB/saylr3	1000	3750	3D finite-element problem from reservoir simulation
HB/sherman4*	1104	3786	3D finite-element problem from oil reservoir modelling
AG-Monien/netz4504	1961	5156	2D finite-element problem
HB/lshp2614	2614	17 980	Triangular finite-element mesh on 2D L-shaped region
AG-Monien/grid2	3276	12 864	2D finite-element problem
HB/saylr4	3564	22 316	$33 \times 6 \times 18$ grid
HB/sherman3	5005	20 033	3D finite-element problem from oil reservoir modelling
AG-Monien/ukerbe1	5891	15 704	2D finite-element problem
AG-Monien/big_dual	30 269	89 858	2D finite-element problem
DNVS/ship_001	34 920	3 777 036	3D finite-element model of ship structure
AG-Monien/brack2	62 631	733 118	3D finite-element problem
DNVS/shipsec8	114 919	3 303 533	3D finite-element model of ship structure
DNVS/fcondp2	201 822	11 294 316	3D finite-element model of oil production platform
INPRO/msdoor2	415 863	19 173 163	Structural problem
Schenk_AFE/af_shell1	504 855	17 562 051	Structural problem

n and nz denote the order of the matrix and the number of entries in the matrix, respectively. * indicates the problem was used in [11].

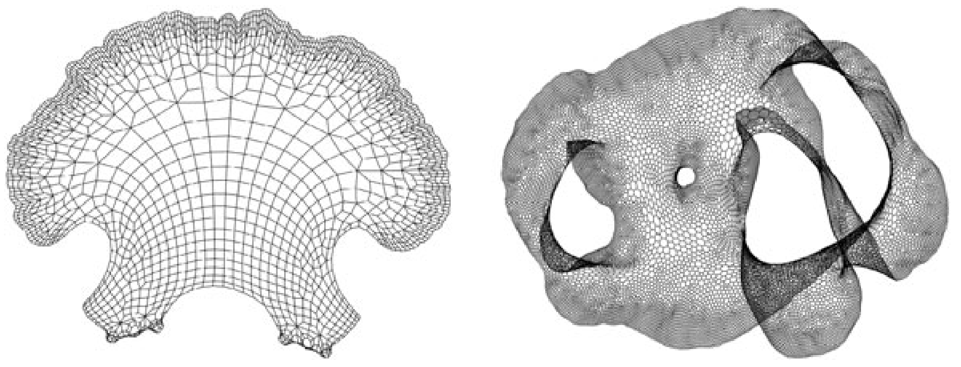

Figure 6. Finite-element meshes for problems AG-Monien/ukerbe1 (left) and AG-Monien/big_dual (right).

4.1. *Results for the* LB *algorithm*

To verify that the LB algorithm performs as expected for mesh problems, in Table II, we present results for the two-dimensional mesh problems used by Duarte *et al.* The starting node *s* is computed using the modified GPS algorithm of Reid and Scott [12]. The optimal solution is also given. We see that, in each case, the LB algorithm computes the optimal maximum antibandwidth or is within two of the optimal. Note that for each mesh, the reordering time was less than 10^{-3} s.

Table III reports results for the practical examples described in Table I, and in the upper part of Figures 7 to 11, the sparsity patterns for some of the problems are plotted both before and after reordering using the level-based algorithm. With the exception of problems AG-Monien/grid1 and AG-Monien/grid2 (which have initial maximum antibandwidths of 12 and 197, respectively), the initial maximum antibandwidth is 1, but there is a large range of values for the initial average antibandwidth. We see that for some of the problems with regular grids of square or cubic elements (such as HB/saylr1 and HB/nos7), *ab* and *av* are increased substantially by relabelling, and the level-based ordering gives the optimal (or close to optimal) antibandwidths (from Equation (2.1), the optimal *ab* for HB/saylr1 is 112, and from (2.2), for HB/nos7, it is approximately 334). However, for other examples (including the DNVS problems, HB/dwt_234 and HB/nos5), whereas the average distance between the diagonal and nearest off-diagonal entry increases (so that *av* increases), the maximum antibandwidth remains small. For some of these latter examples, towards the end of the relabelling, we have to give consecutive labels to nodes that are close to each other in the level-set structure; we can see this in Figure 10 for problem AG-Monien/big_dual. In the case of HB/lshp2614, the grid comprises triangular elements, and as is often the case for triangular meshes, a node in level *r* can have neighbours not only in

Table II. Maximum antibandwidths for two-dimensional mesh problems computed using the LB algorithm.

Problem	LB	Optimal	Problem	LB	Optimal
mesh9×9	36	36	mesh130×7	451	452
mesh50×2	49	49	mesh120×8	476	476
mesh34×3	49	50	mesh110×9	490	491
mesh25×4	47	48	mesh100×10	494	495
mesh20×5	47	48	mesh50×20	489	490
mesh10×10	44	45	mesh40×25	486	488
mesh17×6	47	48	mesh60×17	501	502
mesh13×8	47	48	mesh34×30	494	495
mesh15×7	49	49	mesh80×13	513	514
mesh12×9	49	49	mesh70×15	517	518
mesh11×11	55	55	mesh90×12	533	534
mesh12×12	66	66	mesh33×33	528	528

Table III. Results of applying the level-based (LB) algorithm.

Problem	*ab*	av_{i}	$ratio_{\mathrm{LB}}$
HB/curtis54	4	1.15	10.5
HB/dwt_234	2	2.97	10.6
HB/saylr1	111	1.00	113.2
AG-Monien/grid1	116	82.2	1.45
HB/nos7	330	1.00	340.1
HB/can_445	5	1.19	68.0
HB/nos5	7	2.37	33.8
HB/662_bus	29	45.1	5.44
HB/nos6	329	1.00	330.4
HB/saylr3	1	317.1	1.68
HB/sherman4	257	558.5	1.23
AG-Monien/netz4504	671	18.7	51.7
HB/lshp2614	14	1.00	650.9
AG-Monien/grid2	1626	909.3	1.79
HB/saylr4	1726	1.68	1033.3
HB/sherman3	30	2111.0	1.34
AG-Monien/ukerbe1	2054	57.2	51.7
AG-Monien/big_dual	57	3.54	3400.2
DNVS/ship_001	1	1.00	564.4
AG-Monien/brack2	11	191.8	32.9
DNVS/shipsec8	3	1.00	3886.0
DNVS/fcondp2	5	1.00	8185.0
INPRO/msdoor	1	1.00	14 200.0
Schenk_AFE/af_shell1	488	1.00	27 360.0

The maximum antibandwidth (*ab*) after the LB algorithm is given. The initial average antibandwidth (av_{i}) is also given together with $ratio_{\mathrm{LB}} = av_{\mathrm{LB}}/av_{\mathrm{i}}$, where av_{LB} denotes the average antibandwidth after the LB algorithm.

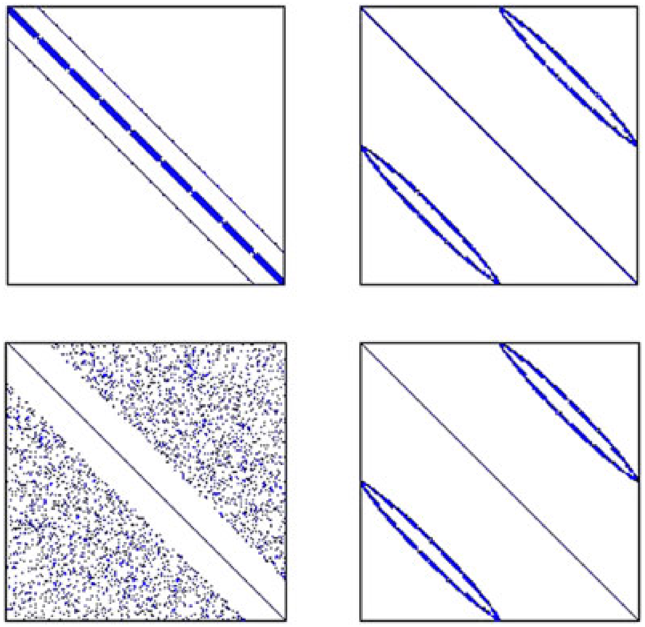
Figure 7. HB/nos7 (upper left), after reordering using the level-based algorithm (upper right), after hill climbing applied to the initial ordering (lower left) and to the level-based ordering (lower right).

l_{r-1} and l_{r+1} but also in l_r, and so we do not have the even and then the odd level set labelling that is possible for square elements. These results show that using the level-based algorithm is not sufficient on its own.

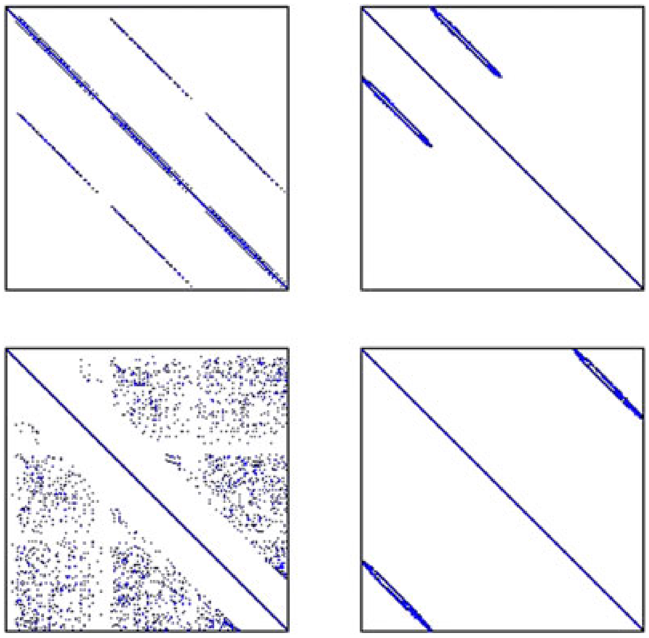
Figure 8. `HB/sherman4` (upper left), after reordering using the level-based algorithm (upper right), after hill climbing applied to the initial ordering (lower left) and to the level-based ordering (lower right).

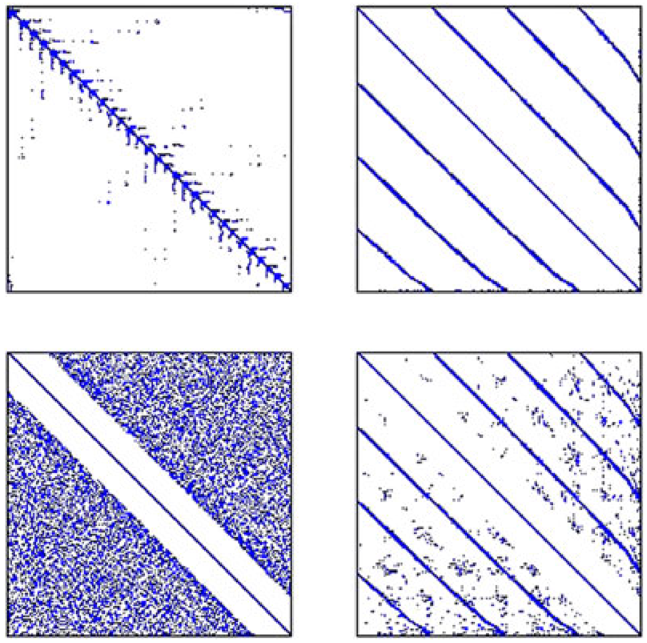
Figure 9. `HB/lshp2614` (upper left), after reordering using the level-based algorithm (upper right), after hill climbing applied to the initial ordering (lower left) and to the level-based ordering (lower right).

We remark that, in the Cuthill–McKee algorithm, the unlabelled neighbours u of $v \in l_{k-1}$ are labelled in order of increasing degree. We tried modifying Algorithm 2 so that the nodes within each level set were pre-ordered by increasing degree. We found that this did not, in general, improve the maximum antibandwidth and, for some problems, it gave poorer results.

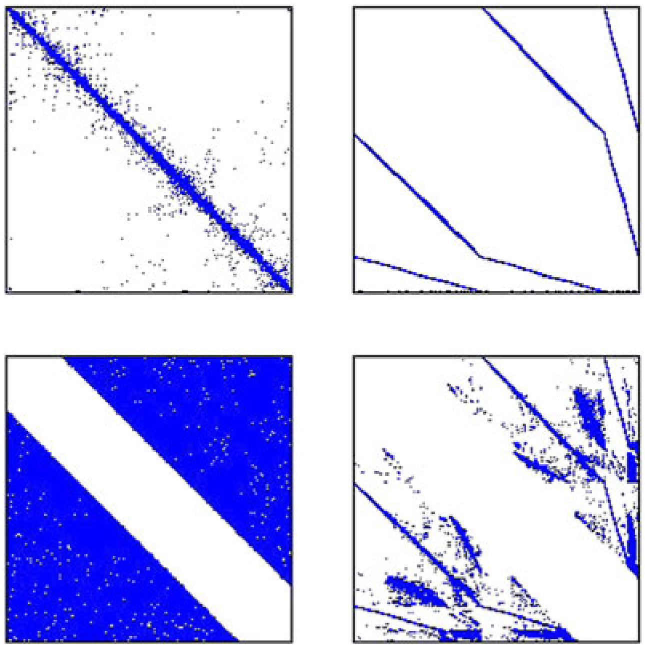
Figure 10. AG-Monien/big_dual (upper left), after reordering using the level-based algorithm (upper right), after hill climbing applied to the initial ordering (lower left) and to the level-based ordering (lower right).

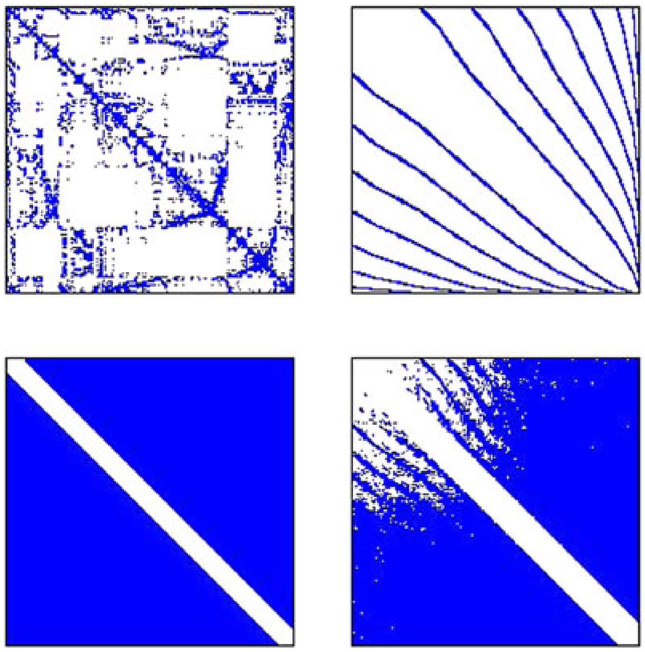
Figure 11. AG-Monien/brack2 (upper left), after reordering using the level-based algorithm (upper right), after hill climbing applied to the initial ordering (lower left) and to the level-based ordering (lower right).

4.2. Hill climbing results

In Table IV, we report results for hill climbing applied to the initial ordering and to the level-based ordering; in the lower part of Figures 8 to 10, the sparsity patterns after hill climbing are plotted for a subset of our test problems. As already noted, for some of the grid problems (and the

Table IV. The maximum antibandwidth (*ab*) after hill climbing is applied to the initial ordering ($\texttt{HC}_\texttt{i}$), after the level-based algorithm (LB) and after hill-climbing follows the level-based algorithm (LB+HC).

	ab			Average antibandwidth ratios		
Problem	$\texttt{HC}_\texttt{i}$	LB	LB+HC	$ratio_{\texttt{HC}_\texttt{i}}$	$ratio_{\texttt{LB}}$	$ratio_{\texttt{LB+HC}}$
HB/curtis54	8	4	**8**	10.5	10.5	10.7
HB/dwt_234	50	2	**80**	28.5	10.6	33.1
HB/saylr1	45	111	**111**	68.4	113.2	113.2
AG-Monien/grid1	70	116	**116**	1.08	1.45	1.45
HB/nos7	105	330	**330**	159.8	340.1	340.1
HB/can_445	46	5	**52**	54.0	68.0	68.0
HB/nos5	32	7	**49**	21.5	33.8	31.8
HB/662_bus	125	29	**163**	4.64	5.44	5.39
HB/nos6	146	329	**329**	208.6	330.4	330.4
HB/saylr3	175	1	**627**	1.62	1.68	2.36
HB/sherman4	168	257	**815**	1.25	1.23	1.72
AG-Monien/netz4504	344	671	**671**	33.3	51.7	51.7
HB/lshp2614	**343**	14	337	460.8	650.9	643.0
AG-Monien/grid2	591	1626	**1626**	0.99	1.79	1.79
HB/saylr4	469	1726	**1726**	448.0	1033.3	1034.5
HB/sherman3	693	30	**3509**	1.27	1.34	1.96
AG-Monien/ukerbe1	1264	2054	**2054**	35.7	51.7	51.7
AG-Monien/big_dual	5760	57	**6645**	2663.0	3400.2	3451.0
DNVS/ship_001	244	1	**319**	347.2	564.4	534.7
AG-Monien/brack2	3480	11	**4984**	30.0	32.9	46.0
DNVS/shipsec8	1664	3	**2246**	2337.0	3886.0	3630.0
DNVS/fcondp2	2921	5	**4020**	4097.0	8185.0	7681.0
INPRO/msdoor	7115	1	**8137**	9849.0	14 200.0	13 910.0
Schenk_AFE/af_shell1	12 970	488	**14 973**	16 980.0	27 360.0	25 990.0

The largest maximum antibandwidth is in bold. The ratios of the average antibandwidths to the original average antibandwidth are also given.

two-dimensional mesh problems), the level-based ordering gives the optimal (or close to optimal) maximum antibandwidths, and so hill climbing cannot improve them further. Comparing columns 2 and 4 of Table IV, we see that, with the exception of HB/lshp2614, applying HC to the level-based ordering gives a larger maximum antibandwidth than applying it to the initial ordering. This illustrates the importance of providing the hill-climbing algorithm with a good initial ordering. As expected, hill climbing applied to the level-based ordering can decrease the average antibandwidth, although the amount by which it decreases is typically less than 5%. We observe that the sparsity patterns that we get by applying hill climbing to the initial ordering and the level-based ordering can be very different, even when the value of *ab* is not too different (for example, AG-Monien/big_dual in Figure 10).

The reverse Cuthill–McKee algorithm is commonly used in place of the Cuthill–McKee algorithm because, although reversing the ordering leaves the bandwidth unchanged, it can reduce the profile [18]. Reversing the LB ordering leaves the maximum antibandwidth and the average antibandwidth unchanged. However, if hill climbing is then applied, the final antibandwidth and average antibandwidth are generally not the same. In our experiments, we found that for the majority of our test cases, the hill climbing results for the reverse LB ordering were poorer than for the LB ordering, and so we do not recommend using the reverse ordering.

4.3. Relaxed hill climbing

In the hill-climbing algorithm, i and j are swapped only if the swap leaves both i and j non-critical. We have investigated whether it can be advantageous to perform a swap even if it results in j becoming critical. The idea is that if hill climbing has stalled (that is, no further swaps can be made to reduce the number of critical entries), we allow a swap that leaves the number of critical entries unchanged in the hope that such a move will lead to later swaps that do reduce the critical

Table V. A comparison of the maximum antibandwidths computed using the standard and relaxed hill-climbing strategies.

Problem	Standard	Relaxed
HB/can_445	52	56
HB/nos5	49	54
HB/lshp2614	337	351
AG-Monien/brack2	4984	5453

entries. We refer to this variant as *relaxed* hill climbing. Note that when implementing this variant, care has to be taken to avoid being into a cycle of swaps that give no gains. Furthermore, the relaxed strategy is only employed once the standard hill climbing strategy has stalled (using it from the start led to much poorer results). Results for relaxed hill climbing are given in Table V (the problems for which the relaxed strategy gave no gain are omitted). We found that the antibandwidth increased for only a few of our test problems and, in general, the additional cost of performing extra searches and swaps for relaxed hill climbing was not beneficial.

4.4. *The effect of random initial permutations*

Finally, we have looked at using the algorithms after applying random symmetric permutations to the given matrix. For the regular two-dimensional mesh problems that comprise our first test set, such permutations have little effect: in all our tests, the computed maximum antibandwidth was again the optimal or within two of the optimal. The results for our more general test set are shown in Table VI. It is indeed the case that if hill climbing is applied directly, better antibandwidths can often be found by considering random permutations. This is because a different initial ordering causes a different sequence of swaps to be performed. We could impose more stringent conditions when a swap is performed to try and reduce sensitivity. The penalty would be further overheads, and in any

Table VI. Best and worst maximum antibandwidths using the given ordering and nine random permutations; where only one number is reported, the best and worst are the same.

Problem	HC_p	LB	LB+HC
HB/curtis54	[5, 8]	[4, 6]	[7, 8]
HB/dwt_234	[36, 50]	2	[48, 80]
HB/saylr1	[42, 54]	[111, 112]	[111, 112]
AG-Monien/grid1	[43, 70]	116	116
HB/nos7	[104, 131]	330	330
HB/can_445	[40, 53]	[1, 5]	[47, 55]
HB/nos5	[32, 43]	[3, 7]	[43, 49]
HB/662_bus	[87, 125]	[4, 29]	[126, 163]
HB/nos6	[114, 146]	329	329
HB/saylr3	[154, 193]	1	[625, 627]
HB/sherman4	[158, 187]	[257, 258]	[815, 817]
AG-Monien/netz4504	[308, 438]	671	671
HB/lshp2614	[315, 359]	[11, 16]	[337, 423]
AG-Monien/grid2	[537, 662]	[1624, 1626]	[1624, 1626]
HB/saylr4	[441, 580]	[1724, 1726]	[1724, 1726]
HB/sherman3	[557, 755]	[29, 30]	[2016, 3509]
AG-Monien/ukerbe1	[971, 1264]	2054	2054
AG-Monien/big_dual	[5267, 6267]	[31, 259]	[6526, 6645]
DNVS/ship_001	[188, 244]	1	319
AG-Monien/brack2	[3025, 3480]	[11, 44]	[4984, 5313]
DNVS/shipsec8	[1552, 1664]	[1, 3]	[2220, 2279]
DNVS/fcondp2	[2703, 3035]	[1, 8]	[3815, 4074]

HC_p is the antibandwidth after hill climbing is applied to the permuted matrix.

case, the sensitivity can be viewed as advantageous as it allows us, by permuting and rerunning, to explore the 'landscape' of the objective function and hopefully to then be close to finding the global optimal antibandwidth.

As we would expect, for many problems, the maximum antibandwidth obtained from the level-based algorithm is not sensitive to the initial ordering. Where there is a difference between the best and the worst LB maximum antibandwidths it is because the pseudodiameter computed by the modified GPS algorithm is dependent on the initial ordering, and this in turn effects the level set structure used by the LB algorithm. Moreover, tie breaking when selecting the nodes within each level can have an effect. Again, we could impose rules regarding tie breaking that would reduce sensitivity.

5. COMPARISONS WITH OTHER APPROACHES

So far, we have presented results for our proposed level-based algorithm with hill-climbing refinement (LB+HC). In this section, we consider the GRASP and GSpectral approaches and perform some comparisons.

GRASP is a well-known metaheuristic that has been successfully applied to many hard combinatorial optimization problems. It is an iterative process, in which each GRASP iteration comprises two phases: construction and local search. The construction phase builds a feasible solution, whose neighbourhood is explored by the local search. The best solution over all GRASP iterations is returned as the result. A detailed description of both the construction and local search phases for the antibandwidth maximization problem is given in Duarte *et al.* [11].

The GSpectral algorithm of Hu *et al.* [10] approximates the antibandwidth maximization problem by finding a permutation such that the sum of the squares of the difference of the row and column indices is maximized, that is,

$$\max_{p\in P} \sum_{a_{ij}\neq 0} (p(i)-p(j))^2.$$

Here, P is the set of all possible permutations of $\{1,2,\dots,n\}$. The vector p is a permutation, and $p(i)$ is the ith element of this vector. By relaxing this problem further so that the row and column labels are real numbers $c(i)$ with a normalization constraint, this problem becomes

$$\max_{c\in R^n} \sum_{a_{ij}\neq 0} (c(i)-c(j))^2, \text{ subject to } \sum_{k=1}^{n} c(k)^2 = 1.$$

It can be shown that c is the eigenvector corresponding to the largest eigenvalue of the Laplacian of the graph induced by the matrix A. Once c has been computed, a row and column permutation is obtained by ordering the entries of c. GSpectral then uses a greedy local refinement algorithm to try and improve the antibandwidth. This refinement proceeds by swapping the labels of pairs of indices, with a swap allowed if it increases the local antibandwidths, that is, a swap between i and j is performed provided that $\min\{ab(i), ab(j)\}$ increases, where $ab(i)$ is the local antibandwidth given by

$$ab(i) = \min_k\{|i-k| : i \neq k \text{ and } a_{ik} \neq 0\}.$$

Swapping continues until no further swaps can be found. Whereas the eigenvector c of the Laplacian can be found in near linear time for a sparse graph using power iterations, the simple swapping procedure can take time quadratic in the dimension of the matrix, making it expensive for large problems. Note that the interest of Hu *et al.* was not in large problems, and so the time for the

Table VII. A comparison of the maximum antibandwidth computed using `GSpectral`, `GRASP` and the `LB+HC` algorithm.

			ab			Time (s)		
Problem	*n*	*nz*	`GSpectral`	`GRASP`	`LB+HC`	`GSpectral`	`GRASP`	`LB+HC`
`HB/curtis54`	54	291	9	**12**	8	0.01	0.59	0.000
`HB/dwt_234`	234	834	77	∗	**80**	0.02	∗	0.000
`HB/saylr1`	238	1128	72	107	**111**	0.02	60.2	0.000
`AG-Monien/grid1`	252	952	77	114	**116**	0.02	57.7	0.000
`HB/nos7`	330	4617	177	324	**330**	0.22	854.9	0.000
`HB/can_445`	445	3809	62	**84**	52	0.15	547.8	0.001
`HB/nos5`	468	5172	64	**66**	49	0.32	257.8	0.002
`HB/662_bus`	662	2474	164	**219**	163	0.13	252.0	0.001
`HB/nos6`	675	3255	212	315	**329**	0.16	736.9	0.000
`HB/saylr3`	1000	3750	249	∗	**627**	0.26	∗	0.010
`HB/sherman4`	1104	3786	274	∗	**815**	0.30	∗	0.006
`AG-Monien/netz4504`	1961	5156	652	∗	**671**	1.03	∗	0.002
`HB/lshp2614`	2614	17 980	**512**	∗	337	5.32	∗	0.002
`AG-Monien/grid2`	3276	12 864	823	∗	**1626**	4.69	∗	0.000
`HB/saylr4`	3564	22 316	873	∗	**1726**	6.29	∗	0.000
`HB/sherman3`	5005	20 033	1246	∗	**3509**	8.93	∗	0.37
`AG-Monien/ukerbe1`	5891	15 704	1990	∗	**2054**	11.92	∗	0.001
`AG-Monien/big_dual`	30 269	89 858	**10 031**	∗	6645	584.5	∗	0.13
`DNVS/ship_001`	34 920	3 777 036	∗	∗	**319**	∗	∗	15.88
`AG-Monien/brack2`	62 631	733 118	∗	∗	**4984**	∗	∗	6.93
`DNVS/shipsec8`	114 919	3 303 533	∗	∗	**2246**	∗	∗	194.0
`DNVS/fcondp2`	201 822	11 294 316	∗	∗	**4020**	∗	∗	102.5
`INPRO/msdoor2`	415 863	19 173 163	∗	∗	**8137**	∗	∗	2806
`Schenk_AFE/af_shell1`	504 855	17 562 051	∗	∗	**14 973**	∗	∗	2376

The largest maximum antibandwidth is in bold. Times in seconds are also given. ∗ indicates that the time limit of 1 h was exceeded.

refinement step was not a major concern for them. We observe also that it is possible to terminate the refinement process before it has converged (perhaps after a prescribed time limit has been reached or once the rate of progress appears to have slowed) but this was not considered by Hu *et al.*

In Table VII, we compare our computed maximum antibandwidths with those obtained using `GSpectral` and `GRASP` (with default settings); computational times are also given. We note that for some examples, the reported time for `LB+HC` is given as zero because the actual time taken is less than 10^{-3} s. The results show that, in some cases, our approach is successful in computing antibandwidths that are competitive with (or are larger than) those from `GSpectral` and `GRASP`, but for other problems, one or both of the latter gives better results. However, for all problems, our algorithm is faster, significantly so on larger problems, than `GSpectral` and `GRASP`. As expected, the `LB+HC` algorithm works well on the mesh-based problems but less well on problems from other application areas (although in all cases, the `LB+HC` algorithm substantially increases the maximum antibandwidth compared with the initial ordering).

The level-based algorithm is very fast, but for the large problems, using hill-climbing adds a significant overhead; indeed, it accounts for almost all of the `LB+HC` time. For example, for problem `DNVS/fcondp2`, the time to run the level-based algorithm is 0.27 s, with the hill-climbing refinement then taking more than 102 s. For even larger problems, hill climbing is expensive, and thus, it may be necessary to consider terminating early (although we have no way of knowing whether or not the algorithm is close to converging). Similarly, for the `GSpectral` algorithm, as anticipated most of the time is in the greedy refinement phase, with the cost of the computation of the eigenvector being almost negligible by comparison. For problem `AG-Monien/big_dual`, the time to compute the eigenvector is 0.14 s, whereas the greedy refinement requires 584 s. As expected, `GRASP` is the most expensive approach, and for many of our test problems, the time limit of 1 h that we imposed for practical reasons was exceeded. Note that the limit was overrun by `HB/dwt_234`, one of the smallest of our test examples. `GRASP` was left running overnight on problem `AG-Monien/grid2` but still did not complete. Thus, it is clear that it does not offer a method that is practical for any but small problems.

6. CONCLUDING REMARKS AND FUTURE DIRECTIONS

In this paper, we have discussed the antibandwidth maximization problem. We have proposed a straightforward but new approach to this practical problem that is based on extending the ideas for the bandwidth minimization problem to the antibandwidth maximization problem. The proposed `LB` method is inspired by the well-known Cuthill–McKee algorithm and uses the level sets of a graph. To try and improve the solution quality, we have employed a local hill-climbing algorithm. Our focus has been on problems with an underlying mesh, and for these problems, we have demonstrated that our methods are able to compute high-quality orderings quickly.

A disadvantage of our current approach is that it is a simple two step approach: given an initial ordering, it computes a level-based ordering and then refines it using hill climbing. Although this gives good results for many of the test problems that have an underlying mesh, it does not always work well on more general classes of problems. This suggests that we need to develop further maximization antibandwidth algorithms for non-mesh problems. In a future study, we plan to explore other techniques that are designed for bandwidth reduction to see if the ideas involved can be modified for the antibandwidth maximization problem. In particular, we will look at using a node-centroid algorithm [3, 16] combined with hill climbing and a multilevel approach (see [19] for a multilevel algorithm for reducing the profile of a symmetric matrix).

Finally, in this paper our aim is to maximize the antibandwidth, and we do this using a level-based ordering. If the goal is to maximize the average antibandwidth, a different ordering may be beneficial. In the literature on profile minimization, an ordering based on the Fiedler vector has been shown to lead to better results compared with the level-based reverse Cuthill–McKee algorithm [20]. We have experimented with using the Fiedler vector to maximize the average antibandwidth; preliminary results show that for some of our test problems, this can work well. We plan to investigate this, and other refinement algorithms, further.

ACKNOWLEDGEMENTS

We gratefully acknowledge the use of the University of Florida Sparse Matrix Collection, maintained and made available by Tim Davis and Yifan Hu. We thank Mauricio Resende for giving us access to the `GRASP` package. We would also like to thank the two anonymous referees whose constructive comments have helped improved the overall presentation of this paper. This work was supported by the EPSRC grant EP/E053351/1.

REFERENCES

1. Cuthill E., McKee J. Reducing the bandwidth of sparse symmetric matrices, Proceedings of the 24th National Conference of the ACM, 1969.
2. Gibbs N. E., Poole W. G., Stockmeyer P. K. An algorithm for reducing the bandwidth and profile of a sparse matrix. *SIAM Journal on Numerical Analysis* 1976; **13**:236–250.
3. Lim A., Rodrigues B., Xiao F. A centroid-based approach to solve the bandwidth minimization problem. *Proceedings of the 37th Hawaii International Conference on System Sciences, IEEE* 2004.
4. Martí R., Laguna M., Glover F., Campos V. Reducing the bandwidth of a sparse matrix with tabu search. *European J. of Operational Research* 2001; **135**:211–220.
5. Nana E. P., Plana I., Campos V., Martí R. GRASP and path relinking for the matrix bandwidth minimization. *European J. of Operational Research* 2004; **153**:200–210.
6. Leung JY.-T., Vornberger O., Witthoff J. D. On some variants of the bandwidth minimization problem. *SIAM Journal on Computing* 1984; **13**:650–667.
7. Yixen L., Jinjiang Y. The dual bandwidth problem for graphs. *Journal of Zhengzhou University* 2003; **35**:1–5.
8. Cappanera P. A survey of obnoxious facility location problems. *TR-99-11*, Dipartimento di Informatica, Universit a di Pisa, 1999.
9. Raspaud A., Schröder H., Sýkora O., Török L., Vrt'o I. Antibandwidth and cyclic antibandwidth of meshes and hypercubes. *Discrete Mathematics* 2009; **309**:3541–2552.
10. Hu Y. F., Kobourov S., Veeramoni S. On maximum differential graph coloring, Proceedings of the 18th International Conference on Graph Drawing (GD'10), 2011; 274–286.
11. Duarte A., Martí R., Resende M., Silva R. GRASP with path relinking heuristics for the antibandwidth problem. *Networks* 2011. doi: 10.1002/net.20418.
12. Reid J. K., Scott J. A. Ordering symmetric sparse matrices for small profile and wavefront. *International Journal on Numerical Methods in Engineering* 1999; **45**:1737–1755.
13. Paulino G. H., Menezes I. V. M, Gattass M., Mukherjee S. A new algorithm for finding a pseudoperipheral vertex or the endpoints of a pseudodiameter in a graph. *Communications in Numerical Methods in Engineering* 1994; **10**:913–926.
14. Miller Z., Pritikin D. On the separation number of a graph. *Networks* 1989; **19**:651–666.
15. Török L., Vrt'o I. Antibandwidth of three-dimensional meshes. *Discrete Mathematics* 2010; **310**:505–510.
16. Reid J. K., Scott J. A. Reducing the total bandwidth of a sparse unsymmetric matrix. *SIAM Journal on Matrix Analysis and Applications* 2006; **28**(3):805–821.
17. Davis T. A., Hu Y. The University of Florida Sparse Matrix Collection. *ACM Transactions on Mathematical Software* 2011; **38**(1). Article 1, 25 pages.
18. George A. Computer implementation of the finite-element method. *STAN CS-71-208, Ph.D Thesis*, Department of Computer Science, Stanford University, 1971.
19. Hu Y. F., Scott J. A. A multilevel algorithm for wavefront reduction. *SIAM Journal on Scientific Computing* 2001; **23**:1352–1375.
20. Kumfert G., Pothen A. Two improved algorithms for envelope and wavefront reduction. *BIT* 1997; **37:3**:559–590.