Science & Technology Facilities Council

Technical Report
RAL-TR-2011-019

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

J Scott, Y Hu

November 2011

RAL Library
STFC Rutherford Appleton Laboratory
R61
Harwell Oxford
Didcot
OX11 0QX

Tel: +44(0)1235 445384
Fax: +44(0)1235 446403
email: libraryral@stfc.ac.uk

Science and Technology Facilities Council reports are available online at: http://epubs.stfc.ac.uk

ISSN 1358- 6254

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

Jennifer Scott[1] and Yifan Hu[2]

ABSTRACT

The antibandwidth maximization problem is 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.

Keywords: antibandwidth maximiziation, sparse matrices.

AMS(MOS) subject classifications: 65F30, 65F50

---

[1] Computational Science and Engineering Department, Rutherford Appleton Laboratory, Chilton, Oxfordshire, OX11 0QX, UK.
Email: jennifer.scott@stfc.ac.uk
Work supported by EPSRC grant EP/E053351/1.

[2] AT&T Labs Research, 180 Park Avenue, Florham Park, New Jersey, NJ 07932, USA.
Email: yifanhu@research.att.com

Current reports available from `http://www.numerical.rl.ac.uk/reports/reports.html`.

October 26, 2011

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

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

(see, for example, [1], [5], [10], [11], [13]). 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

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

In terms of graphs, the antibandwidth 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. [9] in 1984 in connection with multiprocessor scheduling problems. It is also referred to as the *dual bandwidth* problem [19] or the *separation* problem [9]. It arises in a number of practical applications. For example, if the nodes represent sensitive facilities or chemicals, there may be a desire to locate them as far from each other as possible (or at least, further apart than some specified distance). 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 [9]. 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, [15] and the references therein). Recently, there has been an interest in developing algorithms for computing solutions for general graphs that are close to the optimal. In particular, Hu, Kobourov and Veeramoni [6] 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 matrix and then using a greedy refinement algorithm. They have applied this to the maximum differential graph colouring problem and report results for some small examples. Duarte, Martí, Resende and Silva [3] have proposed a linear integer programming formulation and several heuristics based on GRASP (Greedy Randomized Adaptive Search Procedure) with path relinking. They report some high-quality computational results for general matrices, although the run-times for their relatively modest-sized test problems (matrices of order less than 9000) are quite high (typically several minutes for their fastest approach applied to their largest problems). Thus we would like to develop alternative algorithms for the antibandwidth problem that are significantly faster while retaining good quality. This paper is the first step in achieving this aim.

Many algorithms for reducing the bandwidth of a sparse symmetric matrix 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. An important and well-known example of a bandwidth reduction algorithm that uses $\mathcal{G}$ is the Cuthill-McKee algorithm [1] and its many variants, including the Gibbs-Poole-Stockmeyer algorithm [5]. The Cuthill-McKee algorithm constructs a level set structure of $\mathcal{G}$ 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 than can be tackled by either the `GSpectral` or GRASP approaches.

The rest of this paper is organised as follows. We begin (Section 2) by briefly recalling the Cuthill-McKee algorithm and then considering how it might be modified for the antibandwidth problem. In

Section 3, we look at modifying the hill-climbing algorithm of Lim, Rodrigues and Xiao [10] to improve a given ordering. Our proposed algorithms are used to reorder a set of test matrices and our results are compared with those of Duarte et al [3]. We summarise our findings and discuss future work in Section 4.

We end this section by introducing our test problems. Our first set consists of the 24 two-dimensional meshes that are used by Duarte et al. [3]. They are constructed as the Cartesian product of two paths and optimal solutions for the antibandwidth problem are known by construction (see [15]). Our second set (see Table 1.1) is taken from the University of Florida Sparse Matrix Collection [2] and comes from the DNVS, HB and AG-Monien groups. The order n ranges from 54 to 201822. 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, Figures 1.1 and 1.2). Some of the smaller problems were chosen because they appear in the paper by Duarte et al. [3].

Table 1.1: Test problems. n and nz denote the order of the matrix and the number of entries in the matrix. * indicates the problem was used in [3].

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	17980	Triangular finite-element mesh on 2D L-shaped region
AG-Monien/grid2	3276	12864	2D finite-element problem
HB/saylr4	3564	22316	$33 \times 6 \times 18$ grid
HB/sherman3	5005	20033	3D finite-element problem from oil reservoir modelling
AG-Monien/ukerbe1	5891	15704	2D finite-element problem
AG-Monien/big_dual	30269	89858	2D finite-element problem
DNVS/ship_001	34920	3777036	3D finite-element model of ship structure
AG-Monien/brack2	62631	733118	3D finite-element problem
DNVS/shipsec8	114919	3303533	3D finite-element model of ship structure
DNVS/fcondp2	201822	11294316	3D finite-element model of oil production platform

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.

2 Level-based approach to antibandwidth problem

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

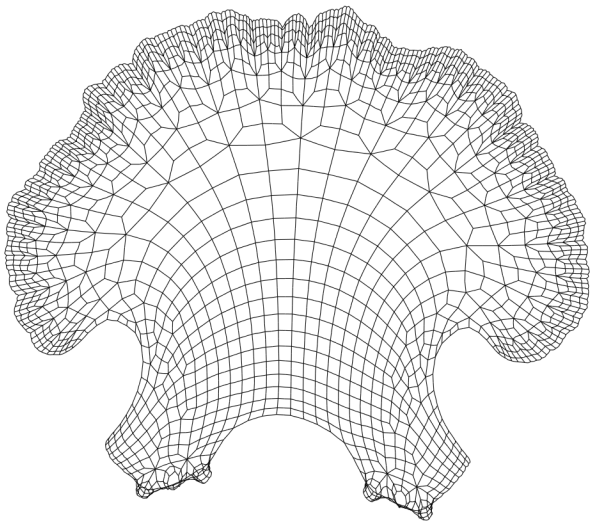

Figure 1.1: Finite-element mesh for problem `AG-Monien/ukerbe1`.

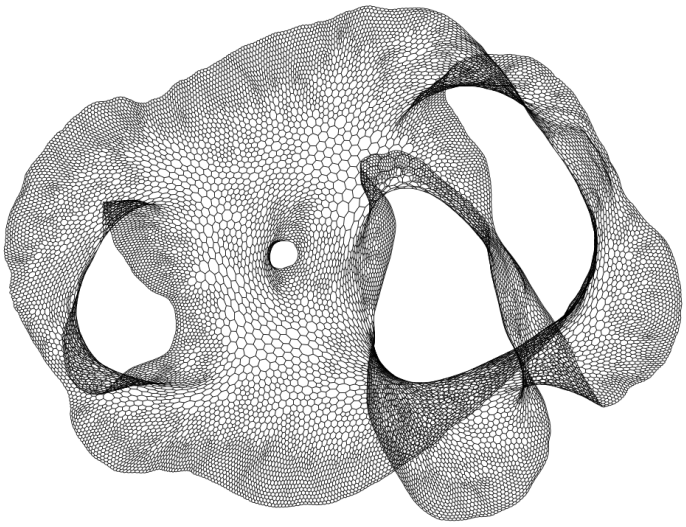

Figure 1.2: Finite-element mesh for problem `AG-Monien/big_dual`.

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 2.1. 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.

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 2.1: Cuthill-McKee ordering algorithm

Ordering the nodes in this way groups them into *level sets*, that is, nodes at the same distance from the starting node. Since 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 $L(s) = \{l_1, l_2, ..., l_h\}$. Algorithms for finding a good starting node are usually based on finding a pseudo-diameter of $\mathcal{G}$ (a pair of nodes that are a maximum distance apart or nearly so). Much effort has gone into efficiently finding a pseudo-diameter; algorithms are generally based either on using level sets (see, for example, [5] and [16] and the references therein) or using the Fiedler vector [14].

2.2 Level-based approach for antibandwidth problem

In their paper, Miller and Pritikin [12] establish tight bounds for the maximum antibandwidth for various classes of graphs. Further bounds were recently presented by Raspaud et al. [15]. In particular, Raspaud et al. showed that for a 2-dimensional $m \times k$ mesh with $m \geq k \geq 2$ 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 describe 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 $\{l_2, l_4, ..., l_h, l_1, l_3, ..., l_{h-1}\}$ (here h is assumed to be even) and ordering nodes in each level set in turn, in natural order.

For a 3-dimensional $m \times m \times m$ mesh Török and Vrt'o [18] 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. For the antibandwidth problem, 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 2.2.

Algorithm 2: Level-based antibandwidth (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**
      $i = i + 1$; label u as node i
      Set $flag(v) = sweep$ for each unlabelled neighbour v of u
    end do
  end do
end do

Figure 2.2: 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 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 having $L(s)$ with one level set (or a small number of level sets) l_r with more entries than the other sets. If the nodes of such an l_r are neighbours, this leads to labelling the nodes of l_r consecutively after all the nodes in the sets l_q with $q \neq r$ have been labelled, resulting 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. To help assess the quality of our labelling, we introduce the *average antibandwidth* to be

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

Note this definition considers upper and lower triangular entries and thus maximizing av_ab is **not** the dual of minimizing the profile of A (which would involve maximizing the sum of the minimum distance from the diagonal in the lower triangular part of A).

To verify that the LB algorithm performs as expected for mesh problems, in Table 2.2 we present results for the 2-dimensional mesh problems used by Duarte et al. The starting node s is computed using the modified GPS algorithm of Reid and Scott [16]. The optimal solution is also given. We see that, in each case, the LB algorithm computes the optimal antibandwidth or is within two of the optimal. Moreover, for each mesh the reordering time was less than 10^{-3} seconds.

Table 2.2: Antibandwidths for 2-dimensional mesh problems computed using the LB algorithm.

Problem	LB	Optimal	Problem	LB	Optimal
mesh9x9	36	36	mesh130x7	451	452
mesh50x2	49	49	mesh120x8	476	476
mesh34x3	49	50	mesh110x9	490	491
mesh25x4	47	48	mesh100x10	494	495
mesh20x5	47	48	mesh50x20	489	490
mesh10x10	44	45	mesh40x25	486	488
mesh17x6	47	48	mesh60x17	501	502
mesh13x8	47	48	mesh34x30	494	495
mesh15x7	49	49	mesh80x13	513	514
mesh12x9	49	49	mesh70x15	517	518
mesh11x11	55	55	mesh90x12	533	534
mesh12x12	66	66	mesh33x33	528	528

Table 2.3 reports results for the practical examples described in Table 1.1 and in the upper part of Figures 2.3 to 2.7 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 antibandwidths of 12 and 197, respectively, the initial antibandwidth is 1. 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_ab 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), while the average distance between the diagonal and nearest off-diagonal entry increases (so that the average antibandwidth increases), the 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 2.6 for problem AG-Monien/big_dual. In the case of HB/lshp2614, the grid comprises triangular elements and, in general, for some r a node has neighbours in each of the level sets l_{r-1}, l_r, and l_{r+1} 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 and it leads us, in the next section, to look for a local search algorithm that may be used to refine the level-based ordering.

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 preordered by increasing degree. We found that this did not, in general, improve the antibandwidth and, for some problems, it gave poorer results.

3 Hill climbing for refinement

Lim et al [10] 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 [17]. In this section, we propose using hill climbing for the antibandwidth problem.

The idea behind hill climbing is that at each step a search is made for a non-critical node to swap with

Table 2.3: The antibandwidth (ab) and average antibandwidth (av_ab) after the level-based algorithm. The initial average antibandwidth is also given.

Problem	ab	av_ab	
		Initial	After LB
HB/curtis54	4	1.148E+00	1.211E+01
HB/dwt_234	2	2.974E+00	3.143E+01
HB/saylr1	111	1.000E+00	1.132E+02
AG-Monien/grid1	116	8.225E+01	1.190E+02
HB/nos7	330	1.000E+00	3.401E+02
HB/can_445	5	1.189E+00	8.084E+01
HB/nos5	7	2.368E+00	7.995E+01
HB/662_bus	29	4.513E+01	2.455E+02
HB/nos6	329	1.000E+00	3.304E+02
HB/saylr3	1	3.171E+02	5.318E+02
HB/sherman4	257	5.585E+02	6.870E+02
AG-Monien/netz4504	671	1.869E+01	9.668E+02
HB/lshp2614	14	1.000E+00	6.509E+02
AG-Monien/grid2	1626	9.093E+02	1.631E+03
HB/saylr4	1726	1.682E+00	1.738E+03
HB/sherman3	30	2.111E+03	2.888E+03
AG-Monien/ukerbe1	2054	5.718E+01	2.959E+03
AG-Monien/big_dual	57	3.541E+00	1.204E+04
DNVS/ship_001	1	1.000E+00	5.644E+02
AG-Monien/brack2	11	1.918E+02	9.317E+03
DNVS/shipsec8	3	1.000E+00	3.886E+03
DNVS/fcondp2	5	1.000E+00	8.185E+03

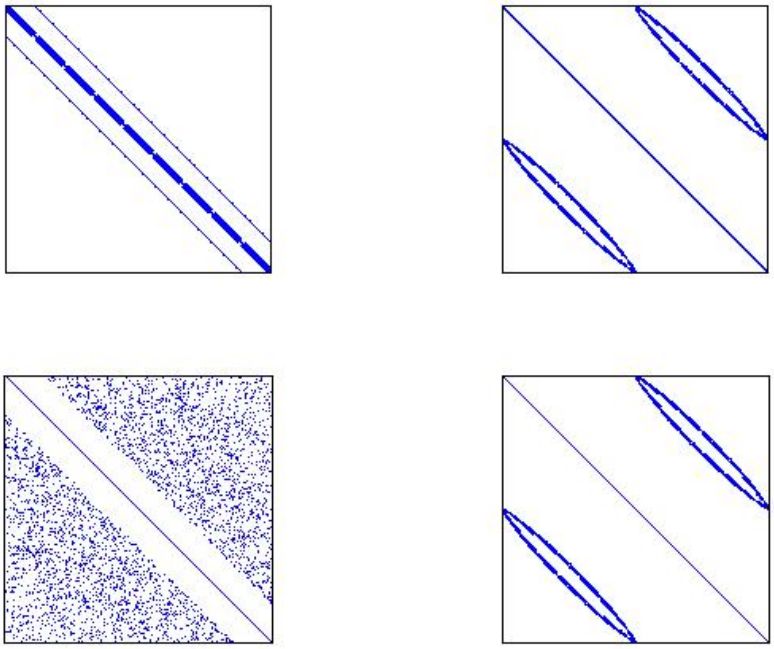

Figure 2.3: `HB/nos7` (upper left), after reordering using the level-based algorithm (upper right), after hill climbing applied to the initial ordering (lower left) and to the level-based ordering (lower right).

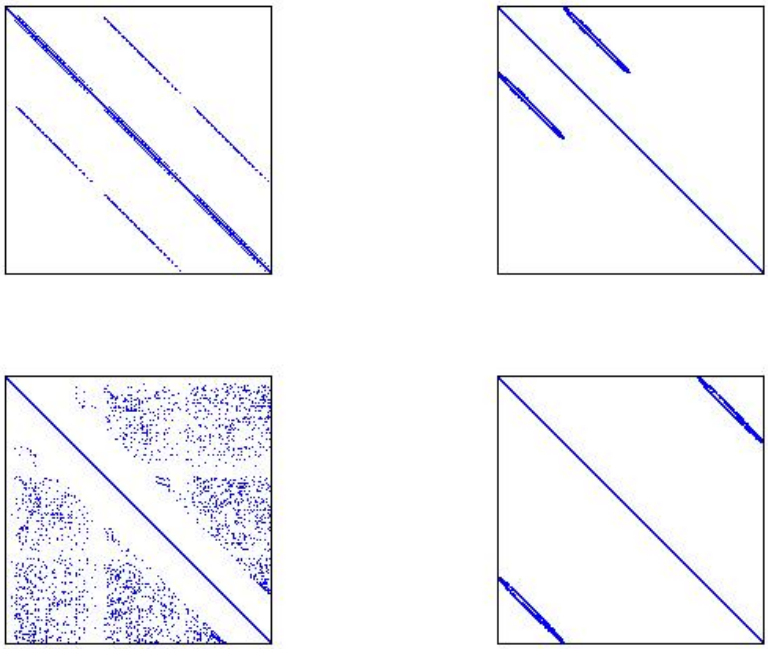

Figure 2.4: `HB/sherman4` (upper left), after reordering using the level-based algorithm (upper right), after hill climbing applied to the initial ordering (lower left) and to the level-based ordering (lower right).

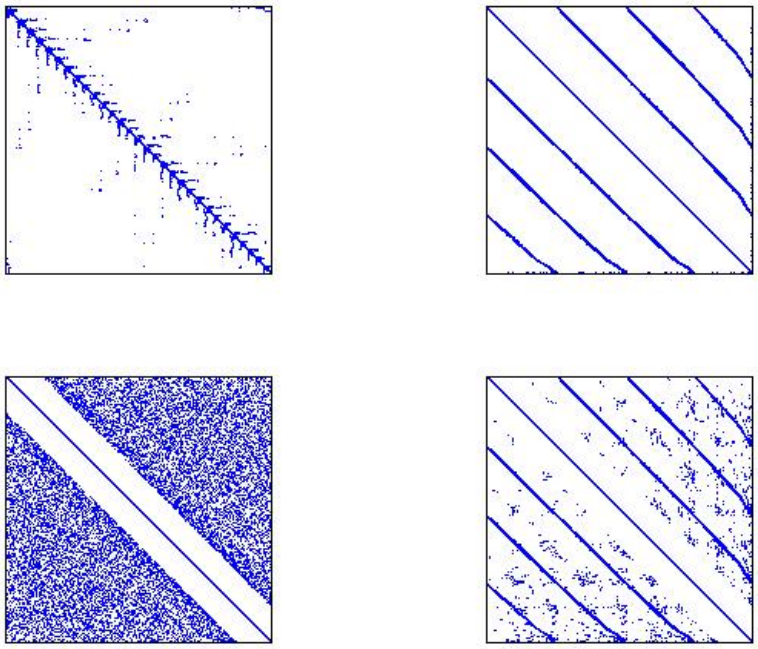

Figure 2.5: `HB/lshp2614` (upper left), after reordering using the level-based algorithm (upper right), after hill climbing applied to the initial ordering (lower left) and to the level-based ordering (lower right).

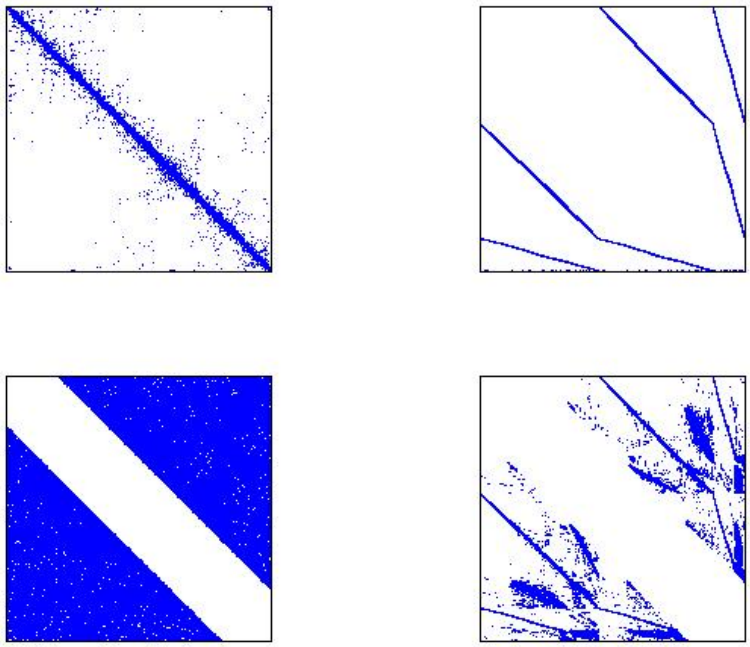

Figure 2.6: `AG-Monien/big_dual` (upper left), after reordering using the level-based algorithm (upper right), after hill climbing applied to the initial ordering (lower left) and to the level-based ordering (lower right).

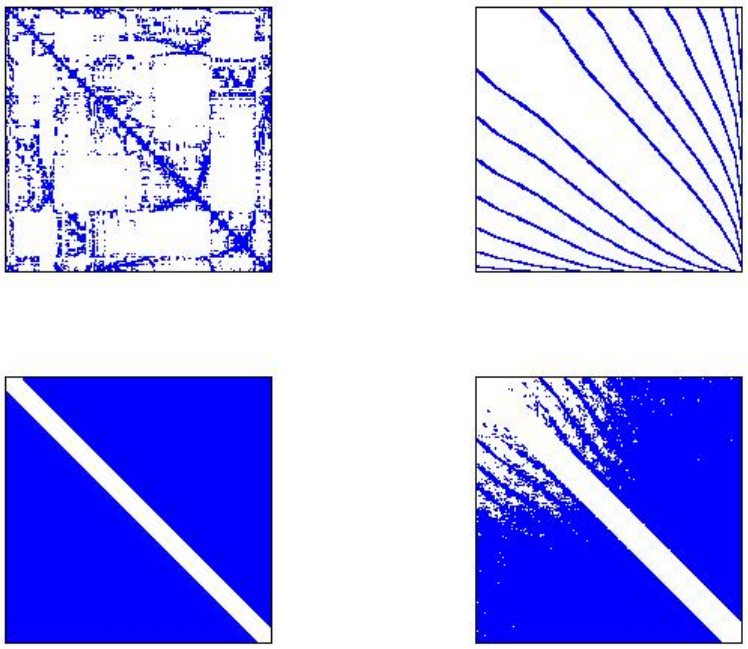

Figure 2.7: `AG-Monien/brack2` (upper left), after reordering using the level-based algorithm (upper right), after hill climbing applied to the initial ordering (lower left) and to the level-based ordering (lower right).

a critical node. For the antibandwidth 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. Since for the antibandwidth problem we need to move entries away from the diagonal, candidates for swapping must be sufficiently far apart. In particular, if row i has a lower critical entry for the current antibandwidth (that is, there is some $k < i$ such that $i - k = ab$) and j lies in the range

$$i - 2 * ab \leq j \leq i - 1$$

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 row i has an upper critical entry, to be a swap candidate j must lie outside the range

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

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 $|l - j| \leq ab$.
3. There exists k such that $a_{kj} \neq 0$ and $|k - i| \leq ab$.

If one of these holds, swapping i and j either decreases the antibandwidth or the number of critical nodes is not reduced. Each accepted swap while the antibandwidth is ab reduces the number of critical entries

by one. If the number of critical entries becomes zero, we restart with antibandwidth $ab + 1$ and repeat the process until none of the critical entries for the current antibandwidth can be swapped to reduce their number. The algorithm is summarized in Figure 3.1. Note that hill climbing cannot decrease the antibandwidth but may decrease the average antibandwidth.

Algorithm 3: 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 3.1: hill-climbing algorithm

The differences between hill climbing for the bandwidth reduction problem and the antibandwidth problem 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 the antibandwidth problem the checking is more expensive since 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 since we found this generally yielded better results.

In Table 3.1, we report results for hill climbing applied to the initial ordering and to the level-based ordering; in the lower part of Figures 2.4 to 2.6 the sparsity patterns after hill climbing are plotted for a subset of our test problems. As already noted, for some of the grid problems, the level-based ordering gives the optimal (or close to optimal) antibandwidths and so hill climbing cannot improve them further. Comparing columns 2 and 4 of Table 3.1, we see that, with the exception of `HB/lshp2614`, applying HC to the level-based ordering gives a larger 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 av_ab, although the amount by which it decreases is typically less than 5 percent. 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 2.6)

Ordering times are given in Table 3.2. These are wall-clock times in seconds computed using the Fortran subroutine `system_clock`. We note that for some examples, the reported time is zero since the actual time taken is less than 10^{-3} seconds. From Table 3.2, it is clear that the level-based algorithm is very fast but for the largest problems using hill-climbing adds a significant overhead.

In practice, for bandwidth reduction, the reverse Cuthill-McKee algorithm is used in place of the Cuthill-McKee algorithm. Reversing the ordering leaves the bandwidth unchanged but can reduce the profile [4]. For the antibandwidth problem, reversing the LB ordering leaves the 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 for the majority of our

Table 3.1: The antibandwidth (ab) and average antibandwidth (av_ab) after hill climbing is applied to the initial ordering (HC_i), after the level-based algorithm (LB) and after hill-climbing follows the level-based algorithm (LB+HC).

	ab				av_ab		
Problem	HC_i	LB	LB+HC	Initial	HC_i	LB	LB+HC
HB/curtis54	8	4	8	1.148E+00	1.209E+01	1.211E+01	1.231E+01
HB/dwt_234	50	2	80	2.974E+00	8.479E+01	3.143E+01	9.856E+01
HB/saylr1	45	111	111	1.000E+00	6.840E+01	1.132E+02	1.132E+02
AG-Monien/grid1	70	116	116	8.225E+01	8.878E+01	1.190E+02	1.190E+02
HB/nos7	105	330	330	1.000E+00	1.598E+02	3.401E+02	3.401E+02
HB/can_445	46	5	52	1.189E+00	6.425E+01	8.084E+01	8.089E+01
HB/nos5	32	7	49	2.368E+00	5.087E+01	7.995E+01	7.518E+01
HB/662_bus	125	29	163	4.513E+01	2.095E+02	2.455E+02	2.450E+02
HB/nos6	146	329	329	1.000E+00	2.086E+02	3.304E+02	3.304E+02
HB/saylr3	175	1	627	3.171E+02	5.152E+02	5.318E+02	7.496E+02
HB/sherman4	168	257	815	5.585E+02	6.972E+02	6.870E+02	9.630E+02
AG-Monien/netz4504	344	671	671	1.869E+01	6.288E+02	9.668E+02	9.668E+02
HB/lshp2614	343	14	337	1.000E+00	4.608E+02	6.509E+02	6.430E+02
AG-Monien/grid2	591	1626	1626	9.093E+02	9.053E+02	1.631E+03	1.631E+03
HB/saylr4	469	1726	1726	1.682E+00	7.526E+02	1.738E+03	1.738E+03
HB/sherman3	693	30	3509	2.111E+03	2.697E+03	2.888E+03	4.146E+03
AG-Monien/ukerbe1	1264	2054	2054	5.718E+01	2.043E+03	2.959E+03	2.959E+03
AG-Monien/big_dual	5760	57	6645	3.541E+00	9.043E+03	1.204E+04	1.222E+04
DNVS/ship_001	244	1	319	1.000E+00	3.472E+02	5.644E+02	5.347E+02
AG-Monien/brack2	3480	11	4984	1.918E+02	5.759E+03	9.317E+03	8.826E+03
DNVS/shipsec8	1664	3	2246	1.000E+00	2.337E+03	3.886E+03	3.630E+03
DNVS/fcondp2	2921	5	4020	1.000E+00	4.097E+03	8.185E+03	7.681E+03

Table 3.2: Times (in seconds) for hill climbing applied to the initial ordering (HC_i), for the level-based algorithm (LB) and for hill-climbing following the level-based algorithm (LB+HC).

Problem	HC_i	LB	LB+HC
HB/curtis54	0.000	0.000	0.000
HB/dwt_234	0.000	0.000	0.000
HB/saylr1	0.001	0.000	0.000
AG-Monien/grid1	0.001	0.000	0.000
HB/nos7	0.002	0.000	0.000
HB/can_445	0.002	0.001	0.001
HB/nos5	0.003	0.001	0.001
HB/662_bus	0.001	0.000	0.000
HB/nos6	0.002	0.000	0.000
HB/saylr3	0.002	0.000	0.011
HB/sherman4	0.002	0.000	0.006
AG-Monien/netz4504	0.003	0.000	0.000
HB/lshp2614	0.018	0.001	0.001
AG-Monien/grid2	0.030	0.001	0.001
HB/saylr4	0.018	0.001	0.001
HB/sherman3	0.038	0.001	0.403
AG-Monien/ukerbe1	0.020	0.001	0.000
AG-Monien/big_dual	0.378	0.007	0.128
DNVS/ship_001	40.38	0.075	15.62
AG-Monien/brack2	3.472	0.030	5.912
DNVS/shipsec8	49.37	0.107	193.6
DNVS/fcondp2	89.90	0.266	102.3

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.

3.1 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, 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 entries. We refer to this variant as *relaxed* hill climbing. Note that when implementing this variant care has to be taken to avoid getting 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 3.3 (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.

3.2 The effect of random initial permutations

Finally, we have looked at using the algorithms after applying random symmetric permutations to the given matrix. The results are shown in Table 3.4. It is indeed the case that, if hill climbing is applied directly, better antibandwidths can often be found in this way. However, as we would expect, for many problems, the antibandwidth obtained from the level-based algorithm is not very sensitive to the initial ordering. Where the best and the worse LB antibandwidths differ it is because the pseudodiameter computed by the

Table 3.3: A comparison of the 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

modified GPS algorithm is different and hence the level set structure is not the same or, in some cases, tie-breaking when selecting the nodes within each level can have an effect.

Table 3.4: Best and worse antibandwidths using the given ordering and nine random permutations; where only one number is reported, the best and worse are the same. HC_p is the antibandwidth after hill climbing is applied to the permuted matrix.

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]

4 Concluding remarks and future directions

So far, we have only presented results for our proposed level-based algorithm with hill-climbing refinement (LB+HC). In Table 4.1, we compare our computed antibandwidths with those obtained using `GSpectral` of Hu et al. [6] and the best reported results of Duarte et al. for their GRASP algorithm. 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. As expected, the LB+HC algorithm works well on the mesh-based problems but less well on problems from other application areas. However, in all cases, the LB+HC algorithm significantly increases

the antibandwidth compared to the initial ordering and the computational times are very much less than those given in Duarte et al. The time taken by `GSpectral` is milli-seconds for the problems in Table 4.1 but when tested on `DNVS/ship_001`, the greedy refinement failed to converge in 90 minutes and thus it is not a practical method for large problems.

Table 4.1: A comparison of the antibandwidth computed using `GSpectral`, the GRASP approach and the LB+HC algorithm.

Problem	`GSpectral`	GRASP	LB+HC
`HB/curtis54`	9	12	8
`HB/dwt_234`	77	51	80
`HB/can_445`	62	85	55
`HB/662_bus`	164	220	163
`HB/nos6`	212	328	329
`HB/sherman4`	274	261	817

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. While 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 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 problem. In particular, we will look at using a node-centroid algorithm [10, 17] combined with hill climbing and a multilevel approach (see [7] 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 aim to 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 to the level-based reverse Cuthill-McKee algorithm [8]. 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 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.

References

[1] E. Cuthill and J. McKee, *Reducing the bandwidth of sparse symmetric matrices*, in Proceedings of the 24th National Conference of the ACM, Brandon Systems Press, 1969.

[2] T. Davis and Y. Hu, *The University of Florida Sparse Matrix Collection*, ACM Trans. Math. Softw., (2011). to appear.

[3] A. Duarte, R. Martí, M. Resende, and R. Silva, *GRASP with path relinking heuristics for the antibandwidth problem*, Networks, (2011). doi: 10.1002/net.20418.

[4] A. George, *Computer implementation of the finite-element method*, Report STAN CS-71-208, Ph.D Thesis, Department of Computer Science, Stanford University, 1971.

[5] N. Gibbs, W. Poole, and P. Stockmeyer, *An algorithm for reducing the bandwidth and profile of a sparse matrix*, SIAM J. Numerical Analysis, 13 (1976), pp. 236–250.

[6] Y. Hu, S. Kobourov, and S. Veeramoni, *On maximum differential graph coloring*, in Proceedings of the 18th international conference on graph drawing (GD'10), Springer-Verlag, 2011, pp. 274–286.

[7] Y. Hu and J. Scott, *A multilevel algorithm for wavefront reduction*, SIAM J. Sci. Comput., 23 (2001), pp. 1352–1375.

[8] G. Kumfert and A. Pothen, *Two improved algorithms for envelope and wavefront reduction*, BIT, 37:3 (1997), pp. 559–590.

[9] J. Y.-T. Leung, O. Vornberger, and J. Witthoff, *On some variants of the bandwidth minimization problem*, SIAM J. on Computing, 13 (1984), pp. 650–667.

[10] A. Lim, B. Rodrigues, and F. Xiao, *A centroid-based approach to solve the bandwidth minimization problem*, Proceedings of the 37th Hawaii international conference on system sciences, IEEE, (2004).

[11] R. Martí, M. Laguna, F. Glover, and V. Campos, *Reducing the bandwidth of a sparse matrix with tabu search*, European J. of Operational Research, 135 (2001), pp. 211–220.

[12] Z. Miller and D. Pritikin, *On the separation number of a graph*, NETWORKS, 19 (1989), pp. 651–666.

[13] E. P. nana, I. Plana, V. Campos, and R. Martí, *GRASP and path relinking for the matrix bandwidth minimization*, European J. of Operational Research, 153 (2004), pp. 200–210.

[14] G. Paulino, I. Menezes, M. Gattass, and S. Mukherjee, *A new algorithm for finding a pseudoperipheral vertex or the endpoints of a pseudodiameter in a graph*, Communications in Numer. Meth. Engng, 10 (1994), pp. 913–926.

[15] A. Raspaud, H. Schröder, O. Sýkora, L. Török, and I. Vrt'o, *Antibandwidth and cyclic antibandwidth of meshes and hypercubes*, Discrete Mathematics, 309 (2009), pp. 3541–2552.

[16] J. Reid and J. Scott, *Ordering symmetric sparse matrices for small profile and wavefront*, Inter. Journal on Numerical Methods in Engineering, 45 (1999), pp. 1737–1755.

[17] ——, *Reducing the total bandwidth of a sparse unsymmetric matrix*, SIAM J. Matrix Analysis and Applications, 28 (2006), pp. 805–821.

[18] L. Török and I. Vrt'o, *Antibandwidth of three-dimensional meshes*, Discrete Mathematics, 310 (2010), pp. 505–510.

[19] L. Yixen and Y. Jinjiang, *The dual bandwdith problem for graphs*, J. of Zhengzhou University, 35 (2003), pp. 1–5.