A Genetic Algorithm with Hill Climbing for the Bandwidth Minimization Problem

Andrew Lim[a] , Brian Rodrigues[b], Fei Xiao[c]

[a] *Department of Industrial Engineering and Engineering Management, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong iealim@ust.hk*
[b] *School of Business, Singapore Management University, 469 Bukit Timah Road, Singapore 259756 br@smu.edu.sg*
[c] *Department of Computer Science, National University of Singapore, 3 Science Drive 2, Singapore 117543 xiaofei@comp.nus.edu.sg*

Abstract

In this paper, we propose an integrated Genetic Algorithm with Hill Climbing to solve the matrix bandwidth minimization problem, which is to reduce bandwidth by permuting rows and columns resulting in the nonzero elements residing in a band as close as possible to the diagonal. Many algorithms for this problem have been developed, including the well-known CM and GPS algorithms. Recently, Marti et al., used Tabu Search and Pinana et al. used GRASP with Path Relinking, separately, where both approaches outperformed the GPS algorithm. In this work, our approach is to exploit the Genetic Algorithm technique in global search while using Hill Climbing for local search. Experiments show that this approach achieves the best solution quality when compared with the GPS algorithm, Tabu Search, and the GRASP with Path Relinking methods, while being faster than the latter two newly-developed heuristics.

Keywords: Bandwidth; Genetic Algorithm; Hill Climbing; Crossover; Mutation

---

1 Introduction

For $A = \{a_{ij}\}$ a symmetric matrix, the matrix bandwidth minimization problem is to find a permutation of rows and columns of the matrix A so as to bring all the non-zero elements of A to reside in a band that is as close as possible to the main diagonal, that is to $Minimize\{max\{|i-j| : a_{ij} \neq 0\}\}$. In the context of graphs, the bandwidth minimization problem can be stated as follows: Let $G(V,E)$ be a graph on n vertices. Label all vertices with different labels from the set $1, 2, ..., n$ so that $f(v)$ is the label for vertex v. Then, with the bandwidth of G defined to be $B_f(G) = Max\{|f(u) - f(v)| : (u, v) \in E(G)\}$, the bandwidth minimization problem is to find a labeling, f, which minimizes $B_f(G)$. Note, that we transform a graph bandwidth problem into a matrix bandwidth problem by using its incidence matrix.

The bandwidth minimization problem for matrices originates from the 1950s (Chinn et al. 1982) when structural engineers first analyzed steel frameworks using computers, while the graph bandwidth problem originated independently at the Jet Propulsion Laboratory at Pasadena in 1962, in the study to minimize the maximum absolute errors of 6-bit picture codes that were represented by edge differences in a hypercube.

The bandwidth minimization problem has been found to be relevant to a wide range of applications. For example, in solving large linear systems, Gaussian elimination can be performed in $O(nb^2)$ time on matrices of bandwidth b, which is much faster than the normal $O(n^3)$ algorithm if $b << n$. Bandwidth minimization is also useful in circuit design and saving large hypertext media. Recently Berry et al.(Berry et al. 1996) applied bandwidth minimization to the information retrieval in hypertext. For other applications, see, for example, (Chinn et al., 1982), (Dueck and Jeffs 1995), (Gibbs et al. 1976), (Luo 1992), (Marti et al. 2001) and (Pinana et al. 2002).

Because of the importance of the bandwidth minimization problem, much research has been carried out in developing algorithms for it. It has been proved to be NP-complete by Papadimitriou (Papadimitriou 1976) and Garey et al. (Garey et al. 1978) have shown that it is NP-complete even if the input graph is a tree whose maximum vertex degree is 3. A number of approximation methods have been proposed to solve it since 1960's. In 1969, the well-known CM algorithm (Cuthill and McKee, 1969) appeared, which used Breadth-First Search to construct a level structure of the graph. By labeling the vertex in the graph according to a level structure, good bandwidth minimization results are achieved in a short time. The GPS algorithm (Gibbs et al. 1976) which was proposed Gibbs, Poole and Stockmeyer in 1976, is also based on level structure. Computational results show that the GPS algorithm is comparable to the CM algorithm in solution quality while being several times faster. For other algorithms developed for the bandwidth minimization problem, see, the survey by Chinn et al. (Chinn et al. 1982), and (Gibbs et al. 1976), (Dueck and Jeffs 1995), (Del Corso and Manzini 1999).

Recently, Marti et al. (Marti et al. 2001) used Tabu Search for the problem. They used a candidate list strategy to accelerate the selection of moves in the neighborhood of the current solution. Extensive experimentation show that their Tabu Search outperforms the best-known algorithms in terms of solution quality in reasonable time. A newly developed GRASP with Path Relinking given by Pinana et al. (Pinana et al. 2002) has been shown to achieve better results than the Tabu Search procedure but with longer running times.

In this work, we propose a Genetic Algorithm (GA) integrated with Hill Climbing to solve the bandwidth minimization problem. Genetic Algorithms (Holland 1975), have been widely used in solving combinatorial optimization problems. The Genetic Algorithm, while having been shown to be especially good in global search, is not well suited for tuned search. In developing a solution for the bandwidth minimization problem, we have therefore combined a Genetic Algorithm with a Hill Climbing Algorithm. Computational Results show that this integrated algorithm performs well. Compared with the classical GPS algorithm, and the newly-developed techniques of Tabu Search and GRASP with Path Relinking, it provides the best solution quality while being faster than the latter two heuristics. Our approach has shown that combining a good global search method with a good local search method can result in high quality solutions.

In next Section 2, the basic structure of our algorithm is presented. Follwoing this, each part of the algorithm is explained in detail in Sections 3,4,5. In Section 6, extensive computational results are given together with analysis which includes setting of parameters, analysis of the efficiency of each part of our algorithm and comparison with past algorithms. The Harwell-Boeing Sparse Matrix Collection is used. Finally, we draw conclusions and provide some direction for future research.

2 The Structure of the Algorithm

As we have mentioned in the introduction, the matrix bandwidth minimization problem has been proved to be NP-complete (Papadimitriou 1976). The search space is N!, where N, the dimension of the matrix, is usually very large. Consequently, the search space for this problem is very large for which brute force can only be implemented for up to $N = 30$. In our heuristic approach to this problem, we have combined GA with Hill Climbing, where GA's are efficient at finding good solution patterns, while Hill Climbing can be exploited to quickly tune solutions to reach local optimum.

In our algorithm, we used the traditional structure for the GA. At the beginning, a initial group of chromosomes which represents different solutions is selected. We then process crossover and mutation on this group of chromosomes to generate new chromosomes. Following this, we apply Hill Climbing to each newly-generated chromosome. As the population of chromosomes is not changed, fittest chromosomes will remain in the next generation. After a number of generations the algorithm will stop and the best result saved. The combined GA with Hill Climbing (GA_HC) Algorithm is described as follows:

GA_HC_ALGORITHM

```
Encode Solutions as Chromosomes
Initialize population
Perform Hill Climbing on the initial population
Count fitness value for the initial population
while $MaximumGeneration$ $\geq$Generation  do
  for $i = 1$ to $PopulationSize$ do
    if $rand(0,1) \leq crossoverrate$ then
      Pick up two chromosomes and perform crossover {two new chromosomes will be
      generated}
    end if
    mutation : perform $rand(0,1) \times Dimension$ swaps on the new chromosome
  end for
  for $i = 1$ to $NumberofNewChromosomes$ do
    perform Hill Climbing on NewChromosome[i]
    Count fitness value for NewChromosome[i]
  end for
  Merge the new chromosomes with old chromosomes
  Select fittest chromosomes from all the chromosomes as the next generation
end while
```

We describe the GA and the Hill Climbing method in detail in the following sections.

3 Genetic Algorithm

As stated, the bandwidth minimization problem in graph form is to find a labeling f which minimizes $B_f(G)$. The labeling for nodes $1, 2, ..., n$ will be result in sequences such as $l_1, l_2, ..., l_n$, which represent solutions to the problem. In the GA we use, we directly encode such solution sequences as chromosomes.

3.1 Initial Population

As mentioned before, many classical algorithms such as CM and GPS are based on the level structure generated by BFS, where vertices with the same depth in the BFS will be on the same level. A level structure is a partition of vertices into levels, L_1,L_2,...,L_k which have the following features (Arany et al. 1971, Marti et al. 2001):

1. Vertices adjacent to a vertex in level L_1 are either in L_1 or L_2.
2. Vertices adjacent to vertex in L_k are in either L_k or L_{k-1}.
3. Vertices adjacent to vertex in L_i(for $1 < i < k$) are in either L_{i-1}, L_i or L_{i+1}.

Given a level structure L, the minimum bandwidth $B_f(G)$, when vertices are labeled sequentially by levels, is bounded as in the following:

$$W(L) \le B_f(G) \le 2W(L) - 1 \tag{1}$$

where $W(L)$ is the cardinality of the largest level with the most vertices in the level structure. Equation 1 shows that we can get a reasonably good solution by labeling vertices in the level structrue sequentially.

In our approach to the problem, we build our initial solutions by performing BFS on the graph from different start vertices. Each time, a vertex is randomly picked as the start node for the BFS and a solution is obtained by labeling vertices sequentially by their levels in the BFS.

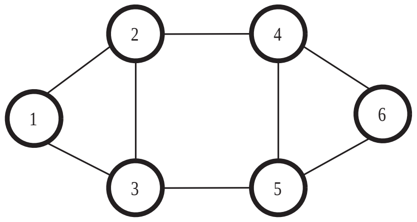

Figure 1: A simple example.

Example: Starting BFS from the vertex 1,4 separately in the graph shown in Figure 1, the vertex sequences in the BFS will be 1,2,3,4,5,6 and 4,2,5,6,1,3, from which we get two initial label sequences: 1,2,3,4,5,6 and 5,2,6,1,3,4.

3.2 Crossover

We used a middle-point crossover as the crossover operator in our GA. The two parent chromosomes will be chosen randomly from the previous generation. The daughter chromosome inherits the first half section directly from the father. As the labels in the sequence cannot be duplicated, the daughter will inherit the labels which have not appeared in its first half section from the mother chromosome in the same order they appeared in the mother chromosome. Similarly, the son chromosome inherits its first-half section completely from the mother and the second-half section from the father. The following example provides detailed description of middle-point crossover.

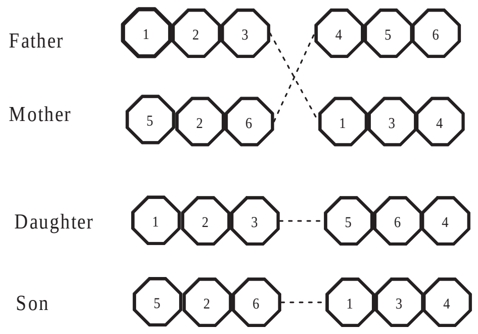

Figure 2: Example of middle-point crossover

As shown in the example in Figure 2, the daughter chromosome inherits its first half-section directly from the father, so its first half section will be 1,2,3. Since 4,5,6 has not appeared in the first half section of the daughter yet, the second half section of the daughter will be composed of the labels 4,5,6 where they should be in the same order as they appeared in the mother sequence. Therefore, we get 5,6,4 as the second half section for the daughter chromosome. In the same way, we get the son chromosome to be 5,2,6,1,3,4.

3.3 Mutation

We used a k-swap mutation scheme in our algorithm. First, a parent chromosome is randomly selected from the previous generation. Then we randomly pick two labels in the parent chromosome and swap them. The swap operator is performed k times. An example for the 2-swap mutation is shown below in Figure 3:

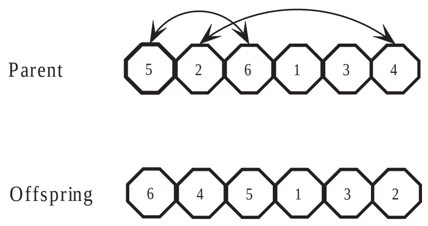

Figure 3: Example of a $k = 2$-swap mutation

As shown in Figure 3, we swapped the first with the third element and the second with sixth element in parent chromosome. The new chromosome has been generated with this 2-swap is 6,4,5,1,3,2.

3.4 Selection

The population for the chromosome group will remain the same for every generation. Each time after generating new chromosomes by crossover and mutation, more chromosomes will be added to the chromosome group. After performing Hill Climbing on the newly generated chromosomes, the bandwidth for the solution the chromosome represents will be obtained. We set the fitness value of the chromosome to be N, the number of vertices in the graph and select chromosomes which have the largest fitness value from the chromosome group which is composed of old chromosomes and newly generated chromosomes as the new generation.

4 Hill Climbing

In the Hill Climbing procedure, we detect critical vertices which are those vertices with the farthest distance from other vertices connected to it, where the distance between vertex u and vertex v is taken to be its label distance, $|f(u) - f(v)|$. We attempt to perform a swap between the label of the current critical vertex with some other vertex to resolve the critical point each time. If we change the label for vertex v, which is a critical vertex, with vertex u, there will be improvement only if vertex v becomes non-critical and vertex u has not changed into a critical vertex (unless it is critical vertex already). The whole process is continued iteratively until no improvement can be obtained.

In the Tabu Search approach proposed by Marti et al. (Marti et al. 2001), the maximum label and minimum label for vertex which connect to vertex v can be defined as :

$$max(v) = max\{f(u) : u \in N(v)\}$$

$$min(v) = min\{f(u) : u \in N(v)\}$$

where $N(v)$ is the set of vertices which connect to vertex v. The best label for v in the current labeling f can be obtained as:

$$mid(v) = \lfloor \frac{max(v) + min(v)}{2} \rfloor$$

Therefore the set of suitable swap vertices for v can be taken as:

$$N'(v) = \{u : |mid(v) - f(u)| < |mid(v) - f(v)|\}$$

which includes all the vertics u with labels $f(u)$ that are closer to $mid(v)$ than $f(v)$. In our approach, we sort the vertices in $N'(v)$ according to the value $|mid(v) - f(u)|$, where the one with lower $|mid(v) - f(u)|$ will be put in front of the sequence. Then we will try to swap the label $f(v)$ of vertex v with the label $f(u)$ of vertex u one by one in the sorted sequence. If the solution quality improves, the labels after a swap will be used as the new solutions. The process is continued until the solution cannot be improved. Our Hill Climbing is simpler than Marti's Tabu Search and can be iteratively run many times.

5 Computational Experiments

We have implemented our GA with Hill Climbing in C and compiled with Microsoft Visual C++ 6.0 on an Intel P4 1.6 GHZ. Experimental results and analysis are divided into five sections. First, in Section 5.1, we test the effect of parameters on the GA with Hill Climbing including the number of generations, the crossover rate and the mutation rate. In Section 5.2, two sets of test data from Harwell-Boeing Sparse Matrix Collection are tested. In Section 5.3, we compare our algorithm with a randomized Hill Climbing Algorithm, demonstrating the effectiveness of the GA as global search method.

5.1 Parameter Setting

We used the matrix bp_1000 from the Harwell-Boeing Sparse Matrix Collection to test the effect of the parameters in the our Genetic Aglorithm. To examine the effect of generations on the GA, we set the population size to be 100, crossover rate to be 1.0 and mutation rate to be 0.002. The result for the matrix bp_1000 is shown in the following Figure 4.

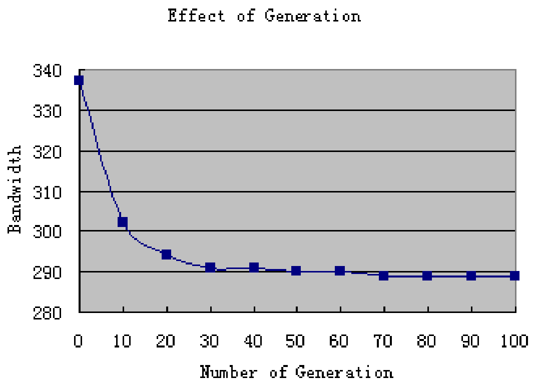

Figure 4: Effect of number of generations

As shown in Figure 4, the bandwidth decreased rapidly for up to 30 generations. After that there are still small decrease of the bandwidth. To balance the quality of the result and the running time, we set the number of generations to be 60 forfollowing experiments.

To examine the effect of the crossover rate, we set the population size to be 100 and the mutation rate to be 0.002. The rate of crossover is selected from {0.1,0.2,0.5,0.8,0.85,0.9,0.95,1.0}. The result is shown in Figure 5.

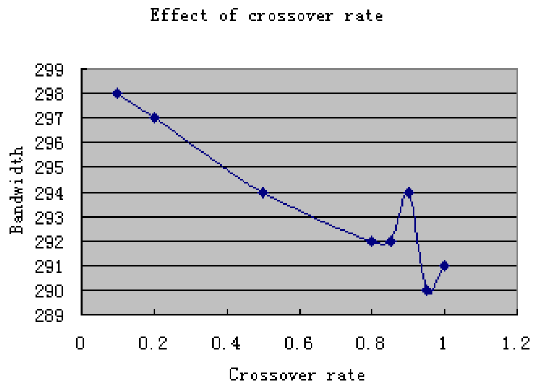

Figure 5: Effect of crossover rate

In Figure 5, we can find the best solutions when the crossover rate is larger than 0.95. As such, we have set the crossover rate to be 1.0 for our algorthm. To test the effect of the mutation rate, we set the population size to be 100. Eight different kinds of mutation rates selected from {0,0.001,0.002,0.005,0.01,0.02,0.05,0.1} are used. The following Figure 6 has shows the results.

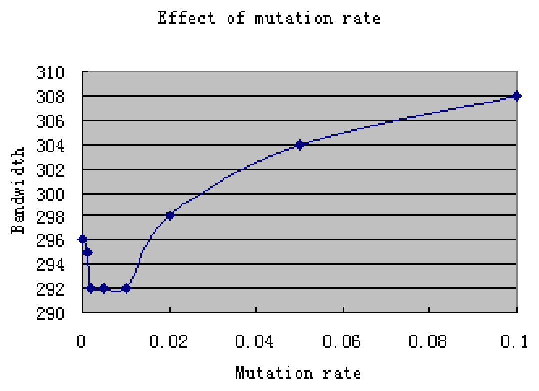

Figure 6: Effect of mutation rate

As shown in Figure 6, when the mutation rate is set to be between 0.002 to 0.005, the minimum bandwidth is obtained. Therefore, we choose the mutation rate to be 0.002 in our algorithm.

Finally, we test the effect of population size. The results are shown in Figure 7.

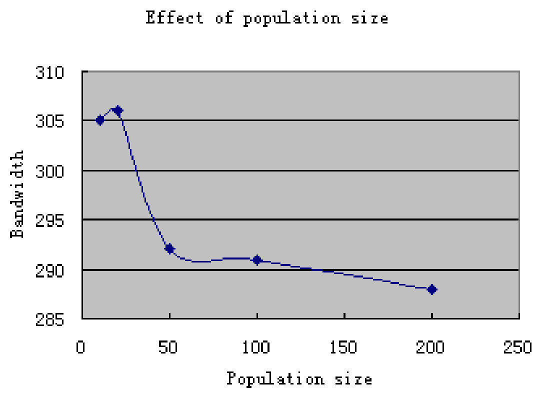

Figure 7: Effect of population size

We find in Figure 7 that the minimum bandwidth decreases as the population size increases. But the running time shown in Figure 8 increases with increasing population size.

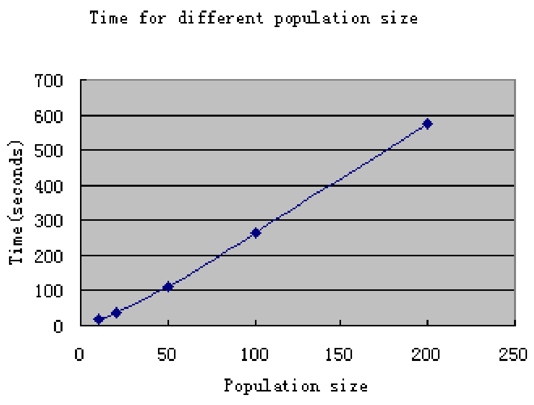

Figure 8: Running time of different population sizes

To balance the solution quality with running time, we have set the population size to be 100.

5.2 Performance of GA with Hill Climbing

We compared our Genetic Algorithm with Hill Climbing (GA_HC) Algorithm with the classical GPS algorithm, the recently developed Tabu Search (TS) and GRASP with Path Relinking (GRASP_PR). The test data we used is same as used by Marti (Marti et al. 2001) and Pinana (Pinana et al. 2002) and are from the Harwell-Boeing Sparse Martix Collection (http://math.nist.gov/MatrixMarket/data /Harwell-Boeing). Two sets of test cases with different dimensions are used. The first test set consists of 33 instances where the dimension of it ranges from 30 to 199. The second test set consists of 80 instances where the dimension of it ranges from 200 to 1000. The following table shows the results of the four methods on the two test sets which include the minimum bandwidth each methods obtained and the running time of each method. We also state the percentage deviation of the other approaches comparing with our GA with Hill Climbing Algorithm.

To compare the speed with the TS and GRASP_PR, we have run the TS and GRASP_PR on same machine PIV 1.6GHZ as we used to run our GA with Hill Climbing. Since the speed for our algorithm can not compete with the GPS, we have compared the result of GPS from Pinana's paper (Pinana et al. 2001) which was run on a machine with K7 1.2 GHZ CPU, approximately 1.3 times slower than our machine.

Table 1: Performance comparison according to problem size

	GPS	TS	GRASP_PR	GA_HC
33 instances with n=30,..,199				
$B_f(G)$	31.42	23.33	22.52	22.48
Deviation	39.77%	3.78%	0.18%	0.00%
CPU seconds	0.003	2.36	4.21	2.54
80 instances with n=200,...,1000				
$B_f(G)$	156.38	100.78	99.43	97.02
Deviation	61.18%	3.88%	2.49%	0.00%
CPU seconds	0.11	121.66	323.19	85.22

We can find in Table 1 that the best solution in quality is obtained for the first test set by the GA_HC which is about 40 percent better than the classical GPS algorithm. The running time for our GA_HC is almost the same the TS while is faster than the GRASP_PR. On the second test set, the GA_HC also obtains the best solution in quality and is faster than the TS and GRASP_PR. Overall, our GA_HC obtains the best solution in quality in shorter time than the newly developed TS and GRASP_PR algorithm.

5.3 Effectiveness of GA

To show the effectiveness of the GA, we used a simple randomly started Hill Climbing to compare with our GA with Hill Climbing. Each time, a labeling sequence is randomly generated, on which we perform Hill Climbing. The following Figure 9 shows the performance between our GA with Hill Climbing and with randomized Hill Climbing.

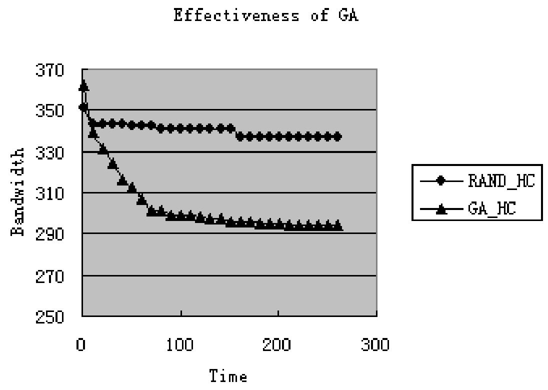

Figure 9: Comparing GA_HC and RAND_HC

From Figure 9, we find that the GA works very well as a guide for the Hill Climbing local search. When we use randomized Hill Climbing, the minimum bandwidth decreases very slowly.

We have also tested the average swap times in the Hill Climbing Algorithm, which is the bottleneck for the speed. In the GA guided Hill Climbing, the average swap time is 534, while in the Randomized Hill Climbing, the average swap time is 2036, which shows that the GA is efficient in finding good solution patterns.

6 Conclusions

We have proposed a combined GA with Hill Climbing for solving the bandwidth minimization problem. Extensive experiments have shown that our new GA with Hill Climbing obtains best solutions for the bandwidth minimization problem in reasonable time. Our approach augments the work recently done by Marti et al. (Marti et al. 2001) and Pinana et al. (Pinana et al. 2002) while throwing light on the use of GA with good local search as a technique for the bandwidth minimization problem.

Alhough our new method gets better solutions than the classical GPS method and is the fastest among newly-developed heuristic methods, the GPS method is still many times faster. Future research may be done to find a faster algorithm with high(er) solution quality.

7 Acknowledgments

We thank Professor Rafael Marti and Professor Vicente Campos, who have kindly provided us with their executable files and the description of the their test data.

References

Arany, I., Smyth, W., F., Szoda, L., 1971, An imporved method for reducing the bandwidth of sparse symmetric matrices, Proc. IFIP Conference, North-Holland, Amsterdam, pp: 1246-1250.

Berry, M., Hendrickson, B., Raghavan, P., 1996. Sparse matrix reordering schedmes for browsing hypertext. In S. Smale J. Renegar, M. Shub, Editor, Lectures in Appl. Math. 32: The Mathematics of Numberical Analysis, pp. 99-123, AMS, Providence, RI.

Chinn, P., Chvatalova, J., Dewdney, A. K., and Gibbs, N. E., 1982, The bandwidth problem for graphs and matrices - a survey. J. Graph Theory, 6:223-254, 1982

Cuthill, E., McKee, J., 1969, Reducing the bandwidth of sparse symmetric matrices. In: Proceed ings of the ACM National Conference, Association for Computing Machinery, New York, pp. 157-172

Del Corso, G. and Manzini, G, 1999, Finding exact solutions to the bandwidth minization problem, Computing, 62, pp. 189-203

Dueck, G.H., Jeffs, J., 1995, A heuristic bandwidth reduction algorithm, Journal of Combinatorial Mathematics and Combinatorial Computing, 18, pp 97-108

Garey, M., Graham, R., Johnson, D., Knuth, D.,1978. Complexity results for bandwidth minimization. SIAM J. Appl. Math., 34:477-495, 1978.

Gibbs, N.E., Poole, W.G., Stockmeyer, P.K., 1976, An algorithm for reducing the bandwidth and profile of sparse matrix. SIAM Journal on Numerical Analysis 13 (2), 236-250.

Glover, F., Laguna, M., 1997, Tabu Search, Kluwer Academic Publishers, Boston.

Holland, J.H., 1975, Adaption in Natural and Artificial Systems, University of Michigan Press, Ann Arbor.

Marti, R., Laguna, M., Glover, F. and Campos, V., 2001, Reducing the Bandwidth of a Sparse Matrix with Tabu Search, European Journal of Operational Research, 135(2), pp.

211-220.

Papadimitriou, C.H., 1976, The NP-completeness of the bandwidth minimization problem. Computing, 16:263-270.

Pinana, E., Plana, I., Campos, V. Marti, R., 2002, in print, European Journal of Operational Research. http://www.uv.es/ rmarti/.