A Centroid-based approach to solve the Bandwidth Minimization Problem

Andrew Lim
Department of Industrial Engineering and Engineering Management, Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
iealim@ust.hk

Brian Rodrigues
School of Business
Singap ore Management University, 469 Bukit Timah Road
Singapore 259756
br@smu.edu.sg

Fei Xiao
School of Computing
National University of Singapore
3 Science Drive 2
Singapore 117543
xiaofei@comp.nus.edu.sg

Abstract

We propose a Node Centroid method with Hill-Climbing to solve the well-known matrix bandwidth minimization problem, which is to permute rows and columns of the matrix to minimize its bandwidth. Many heuristics have been developed for this NP-complete problem including the Cuthill-McKee (CM) and the Gibbs, Poole and Stockmeyer (GPS) algorithms. Recently, heuristics such as Simulated Annealing, Tabu Search and GRASP have been used, where Tabu Search and the GRASP with Path Relinking have achieved significantly better solution quality than the CM and GPS algorithms. Experimentation shows that the Node Centroid method achieves the best solution quality when compared with these while being much faster than the newly-developed algorithms. Also, the new algorithm achieves better solutions than the GPS algorithm in comparable time.

Keywords: sparse matrices, bandwidth, heuristics,

1 Introduction

For $A = \{a_{ij}\}$, a symmetric matrix, the matrix bandwidth minimization problem is to find a permutation of rows and columns of 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 $Minimize\{max\{|i - j| : a_{ij} \neq 0\}\}$.The bandwidth minimization problem also can be stated in the context of graph as: Let $G(V, E)$ be a graph on n vertices. Label, through a function $f : V \rightarrow \{1, 2, ..., n\}$, the vertices of G. Then, with the *bandwidth* of G defined to be $B_f(G) := Max_{(u,v)\in E(G)}|f(u)-f(v)|$, 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 adjacency matrix. The bandwidth minimization problem originates from the 1950s and was proved to be *NP*-complete by Papadimitriou [16]. Garey et al. [7] has shown that it is *NP*-complete even if the input graph is a tree whose maximum vertex degree is 3. The bandwidth minimization problem is relevant to a wide range of optimization applications. In solving large linear systems, Gaussian elimination can be performed in $O(nb^2)$ on the matrices of bandwidth b, which is much faster than the normal $O(n^3)$ algorithm when $b << n$. Bandwidth minimization has also found applications in circuit design and saving large hypertext media. Other practical problems are found in data storage, VLSI design and network survivability. Yet another applications are in industrial electromagnetics [9], finite element methods for approximating solutions of partial differential equations, large-scale power transmission systems, circuit design, chemical kinetics and numerical geophysics [15][17]. In 1969, the classical CM [4] algorithm appeared, which used Breadth-First Search to construct a level structure of the graph. By labeling vertices according to a level structure, good results are achieved in a short time. The GPS algorithm [8],which is also based on the level structure, is comparable with the CM algorithm, but is about eight times faster. Esposito et al. [9] proposed a new WBRA (Wonder Bandwidth Reduction Algorithm), which achieves better results than the CM and GPS algorithm. There are also some other approximation algorithms such as, [6; 14; 3]. A detailed suvey can be found in [5].

In 2000, Marti et al. [15] proposed a new Tabu Search method in which candidates list strategy was used to accelerate the selection of move in a neighborhood. Extensive experimentation showed that their Tabu Search outperformed best-known algorithms in terms of solution quality. In 2002, Pinana et al. [17] used a GRASP (Greedy Randomized Adaptive Search Procedure) with Path Relinking method for the problem. Computational results showed that the GRASP with Path Relinking (GRASP_PR) achieved the best solution quality, although it is slower than Tabu Search. Recently, Genetic Algorithm, Node Shift method and Partical Swarm Optimization have been applied to solve the bandwidth minimization problem by us [11; 12; 13], which have obtained better solution quality comparing with past methods.

In this paper we propose a Node Centroid adjustment method with Hill Climbing, which achieved futher improvement in solving the bandwidth minimization problem. Experimentation shows that this algorithm outperforms the other algorithms in solution quality. Further, a fast version of the algorithm is comparable with CM and GPS algorithm in speed, being about 100 times faster than the newly-developed GRASP with Path Relinking algorithms. In the next Section

2, we give the general framework of the algorithm. In Section 3, we describe the Node Centroid adjustment method in more detail, while in Section 4 we describe the Hill Climbing component of the algorithm. Computational results are reported in Section 5 before we conclude.

2 The Node Centroid with Hill-Climbing algorithm

The Node Centroid method with Hill Climbing (NCHC) employs the strategy of using the Node Centroid method for global search with Hill-Climbing in local search. An initial labeling is generated by performing Breadth-First Search (BFS) on the given graph representation of the matrix with random start vertex. We then use the Node Centroid method to adjust vertices to a central (centroid) position among its neighbors. From this, a new labeling is created on which we perform Hill Climbing to obtain local optima. The Node Centroid method and Hill Climbing are iterated a number of times, following which a new initial labeling is generated by BFS. The entire process is repeated several times within the NCHC algorithm which is described in Algorithm 1 given below. .

Algorithm 1 NCHC

```
for i = 1 to restart_Times do
   Initialize(labeling)
   for j = 1 to NC_Times do
      NC(labeling)
      if j mod 2 ≡ 1 then
         HC(labeling)
      end if
   end for
end for
```

In the algorithm, the NC component comprises of Node Centroid labeling adjustments and HC denotes Hill Climbing. These will be described in more detail in the following sections. The HC procedure is invoked only every other time we perform the NC since it is the bottleneck for the speed of the algorithm and since experimentation has shown that this frequency proportion works well. In the next Section 3, we describe the Node Centroid adjustment method and the generation of initial labelings.

3 The Node Centroid Method

3.1 *Generating Initial Labelings*

As is in many heuristic algorithms, good initial solutions often lead to high quality solutions. In our approach, we achieve this by continuing to use classical level structure in generating initial solutions. Many well-known 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 [1; 15]

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 range:

$$|L| \leq B_f(G) \leq 2|L| - 1 \qquad (1)$$

where $|L|$ is the cardinality of the largest level, L, 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 structure 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 vertex for the BFS and a solution is obtained by labeling vertices sequentially by their levels in the BFS.

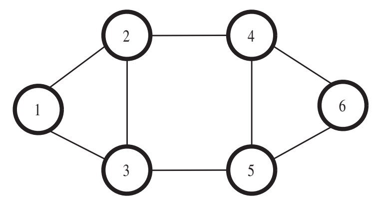


Figure 1: A simple example of BFS

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 *The Node Centroid algorithm*

With the bandwidth of a graph $G(V, E)$ defined by $B_f(G) = Max_{(u,v)\in E(G)}|f(u) - f(v)|$, we define a *vertex neighborhood diameter,* for each vertex v, by $diam_f(v) = Max_{u\in N(v)}|f(u)-f(v)|$, where $N(v) := \{u \in V : (u, v) \in E(G)\}$

Definition 1 For a given labeling, f, the critical value of any vertex, v, is defined as follows:

$$C(v) = \begin{cases} 0 & \text{when } diam_f(v) < B_f(G), \\ 1 & \text{when } diam_f(v) = B_f(G), \end{cases}$$

Further, if $C(v) = 1$,we say that v is *critical*. In our approach, we also consider vertices which are nearly critical by defining vertex v as $\lambda-critical$, whenever $diam_f(v) > \lambda B_f(G)$, for $\lambda \in [0, 1]$. Those $\lambda-$ critical vertices for λ close to 1 would be nearly critical. The set of vertex v's $\lambda - farthest\ neighbors$, $FN_\lambda(v)$, can be defined as:

$$FN_\lambda(v) = N(v) \cap \{u : |f(u) - f(v)| \geq \lambda.B_f(G)\}$$

and the $\lambda - farthest\ neighbor\ bundle$ of vertex v to be $b_\lambda(v) := FN_\lambda(v) \cup \{v\}$. In our approach, we attempt to reduce the neigborhood diameters of $\lambda - critical$ vertices for λ values close to, and including, 1. Here, Node Centroid adjustments are aimed at reducing the diameters of $\lambda - critical$ vertices by attempting to move each toward the centroid of its bundle. To achieve this, we give each vertex a weight $\omega(v)$, which is set according to:

$$\omega(v) = \frac{\sum_{u \in b_\lambda(v)} f(u)}{|b_\lambda(v)|}$$

All the vertices are then sorted into a non-decreasing sequence according to these weights and relabeled. The NC procedure can be described as follows.

Algorithm 2 procedure NC

for $i = 1$ to number_of_vertex **do**
  $w(i) = f(i)$ ($w(i)$ is the weight for vertex i)
  $c(i) = 1$ ($c(i)$ is the number of vertices in the bundle of vertex i)
end for
for $i = 1$ to number_of_edges **do**
  if $|f(u) - f(v)| \geq \lambda.B_f(G)(u, v$ are the two vertices of Edge$[i]$) **then**
    $w(u) = w(u) + f(v)$; $c(u) = c(u) + 1$;
    $w(v) = w(v) + f(u)$; $c(v) = c(v) + 1$;
  end if
end for
for $i = 1$ to number_of_vertex **do**
  $w(i) = \frac{w(i)}{c(i)}$
end for
sort vertices according to their w(weight)
label vertices from 1 to n one by one in the sortied sequence

4 Hill Climbing

In last section, we define the critical value for vertices. For a given labeling, f, we will always have $diam_f(v) \leq B_f(G)$. In order to implement Hill Climbing, for a vertex, v, we define $C(v) = 2$, whenever $diam_f(v)$ becomes larger, through vertex swaps, than the bandwidth before the swap. The Hill Climbing procedure is designed to increase solution quality by swapping vertex labels. In order to judge solution quality, we need to do more analysis.

Definition 2 An edge $e(u, v) \in E$ is said to be a *critical edge* whenever $|f(u) - f(v)| = B_f(G)$.

In Hill Climbing, we measure the solution quality by the current bandwidth of the graph as well as the number of critical edges that are present. When the bandwidth has reduced or, the bandwidth has not changed but the number critical edges has been reduced, the solution quality would have improved. In the Hill Climbing procedure, we detect the critical vertices where $C(v) = 1$ and attempt to perform swaps between the label of the current critical vertex v with some other vertex to reduce the current bandwidth or the number of critical edges. In the Tabu Search approach proposed by Marti et al. [15], the maximum label and minimum label for vertex which connect to vertex v is defined as :

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

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

The best label for v in the current labeling, f, is obtained as:

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

Suitable swap vertices for v were taken to be given by:

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

which includes all the vertices 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 ones with a lower $|mid(v) - f(u)|$ will be put ahead in the node sequence. We then attempt to swap the label $f(v)$ of vertex v with the label $f(u)$ of vertex u one by one in a sorted sequence. If the solution quality improves, the labels after a swap will be used as the new solutions. We detect whether or not the number of critical edges has decreased by the bandwidth before the swap. If these have decreased to 0, then the bandwidth must have been reduced. We take the resultant bandwidth as the new bandwidth. Also, we forbid swaps which increase bandwidth. Theorem 1 below provides us a simple way to detect whether the solution quality has improved. We need the next Lemma first.

Lemma 1 *A critical vertex v can become not critical iff the vertex v is on a critical edge with only one other critical vertex*

Proof: A critical vertex label $f(v)$ can only achieve the distance $B_f(G)$ with $min(v)$ and $max(v)$. If both $min(v)$ and $max(v)$ achieve the distance $B_f(G)$ with $f(v)$, we have $f(v) = \frac{min(v)+max(v)}{2}$, for which changing the label for v can only worsen $B_f(G)$. Conversely, if v is on a critical edge with one other critical vertex, then changing v with the $mid(v)$ makes it not critical.

Theorem 1 *For a given labeling, f, and vertices u, v, by swapping $f(u)$ with $f(v)$, $B_f(G)$ is reduced or the number of critical edges is reduced with $B_f(G)$ unchanged iff the following inequalities are satisfied.*

$$\begin{cases} C'(u) \leq C(u) \\ C'(v) \leq C(v) \\ C'(u) + C'(v) < C(u) + C(v). \end{cases}$$

where $C'(u)$ is the critical value for vertex u after swap comparing with old $B_f(G)$ [1]*,*
where $C'(v)$ is the critical value for vertex v after swap comparing with old $B_f(G)$.

[1]If $B_f(G)$ has been reduced, we will update it only after swapping the two labels

Proof outline If both u, v are not critical, swapping them will not improve the solution quality. Since $C'(u) + C'(v) \geq C(u) + C(v)$, the assertion in the theorem holds. If one of u, v is critical vertex, suppose it is u, so that $C(u) = 1$. By the inequalities, we know that $C'(u) = 0$, which means the number of critical edges is reduced. Conversely, changing u from critical to not critical only can reduce the number of critical edges by 1 by Lemma 1, so vertex v cannot become critical. Hence the inequalities are satisfied. Lastly, if both vertices u and v are critical, where $C(u) = 1$ and $C(v) = 1$, by the inequalities, $C'(u)$ or $C'(v)$ must be 0. If we suppose $C'(u) = 0$, then the number of critical edges is reduced by 1. If $C'(v) = 0$, then the number of critical edges has reduced. If $C'(v) = 1$, as the label for vertex has changed to be $f(u)$, the vertex v only can have one critical edge with $min(v)$ or $max(v)$, otherwise there can only be one other critical label ($\frac{min(v)+max(v)}{2}$). Hence the solution quality has been improved. Conversely, when solution quality is improved, $C'(u)$ or $C'(v)$ must be 0, as both can only connect to one other critical vertex by Lemma 1 and therefore all the inequalities are satisfied. $\triangle$

We use Theorem 1 in the Hill Climbing procedure to count the critical values of the swapped vertices. The whole swap process is iteratively run until the solution quality cannot be improved any more. The HC procedure (Algorithm 3) is described below.

Algorithm 3 procedure HC

```
while can_improve do
  can_improve=false
  for v = 1 to Number_of_vertex do
    if C(v) = 1 then
      for all u such that u ∈ N'(v) do
        if C'(u) ≤ (C(u))∧(C'(v) ≤ (C(v))∧(C'(u)+
        C'(v)) < (C(u) + C(v)) then
          swap(f(v), f(u))
          can_improve=true
          break
        end if
      end for
    end if
  end for
end while
```

5 Computational Results

5.1 Benchmark Results

We have compared our new NCHC algorithm with algorithms developed by other researchers on three sets of test cases from the *Harwell-Boeing Sparse Matrix Collection* (http://math.nist.gov/MatrixMarket/data/Harwell-Boeing/) of standard test matrices, which represent a large spectrum of scientific and engineering applications. These are used as test sets since recent experiments by other researchers have used this test set which is comprehensive and representative of real-world data. We have also developed a fast version of our NCHC algorithm, denoted FNCHC. This is done by allowing paratmeters to be automatically adjusted according to matrix size. The iterating times of the Node Centroid is adjusted according to the number of non-zero entries in the matrix. Though reducing the iterating times of the Node Centroid has decreased the solution quality, the solution is still quite good comparing with past methods. And the whole algorithm is much faster than the old one. In our experiments, we examine the critical factor λ parameter for the FNCHC since experimentation with $\lambda < 1$ in the NCHC gives long running times for which the trade-off with good solution quality is not justified. We use the 80 instances test cases from [17] for λ in the range $[0, 1]$ at intervals of 0.1. The results are shown in Table 1.

Table 1: Critical factor λ for FNCHC

λ	0.1	0.2	0.5	0.6	0.7
$B_f(G)$	103.39	103.18	104.22	103.21	102.75
CPU seconds	1.44	1.80	2.58	2.30	2.28
λ	0.8	0.85	0.9	0.95	1
$B_f(G)$	102.84	103.08	102.83	102.68	106.09
CPU seconds	2.43	2.30	2.10	1.85	0.78

As the results show, for this test set, $\lambda = 0.95$ achieves the best solution quality whereas $\lambda = 1$ gives the best time. This result is interesting providing for the fact that adjusting for those $\lambda - critical$ nodes for values of λ close to 1 results in better results than if we were to only adjust the critical vertices alone. The fact that $\lambda = 1$gives the best time is obvious. For the range of test sets we attempt to balance solution quality with running time choosing λ to be 0.95. For the NCHC algorithm we set λ to be 1.00 since the times taken outweigh the benefits of solution quality disproportionately. Parameters we set for the three test sets are shown in Table 2.

Table 2: Parameters for NCHC and FNCHC

	Test Set 1	Test Set 2	Test Set 3
NCHC			
(No. of Restarts)	Dim	100	100
(No. of NC)	100	200	200
FNCHC			
(No. of Restarts)	5	5	5
(No. of NC)	f(NE)	f(NE)	f(NE)

In table 2, dim is the dimension of the matrix, f(NE) is the function for NE, which is defined in the following:

$$f(NE) = 200/2^{log_{10}(E)-1};$$

where NE is the number of non-zero entries in the matrix.

The first two test sets have also been used in [17]. Experimental results for the two test sets of total 113 instances when compared with the classical GPS [8], the Tabu Search [15] , the GRASP with Path Relinking [17], the Genetic Algorithm and the Node Shift Method we proposed [11; 12; 13] are shown in Table 3, where the dimension for the first 33 instances range from 30 to 199, and the dimensions for the 80 instances set is from 200 to 1000.

Table 3: 113 instances from Harwell-Boeing Collection

	GPS	TS(Marti)	GRASP_PR	NCHC
33 instances				
$B_f(G)$	31.42	23.33	22.52	22.39
CPU seconds	0.003	2.36	4.21	3.37
80 instances				
$B_f(G)$	156.38	100.78	99.43	97.99
CPU seconds	0.11	121.66	323.19	40.20
	FNCHC	GA	NS	PSO
33 instances				
$B_f(G)$	22.79	22.48	22.36	23.21
CPU seconds	0.45	2.54	2.18	2.32
80 instances				
$B_f(G)$	102.68	97.02	97.61	99.96
CPU seconds	1.85	85.02	241.80	96.12

Here, all the metohds are tested on the Intel P4 1.6G CPU except the GPS is tested on AMD K7 1.2G CPU. The P4 1.6G CPU is about 1.33 times faster than the K7 1.2G CPU. As shown in table 3, best solutions in quality are achieved by the NCHC, GA and NS, where NCHC is the fastest one. Meanwhile our FNCHC achieves very good solutions in a short time, which is more than 100 times faster than the GRASP_PR. Although the FNCHC is slower than the GPS algorithm, it obtains solutions 37% and 60% better than the latter. We also compared our approach with the Esposito's WBRA and TS [9; 10] on the DWT test set from the Harwell-Boeing Sparse Matrix Collection for which the results are shown below.

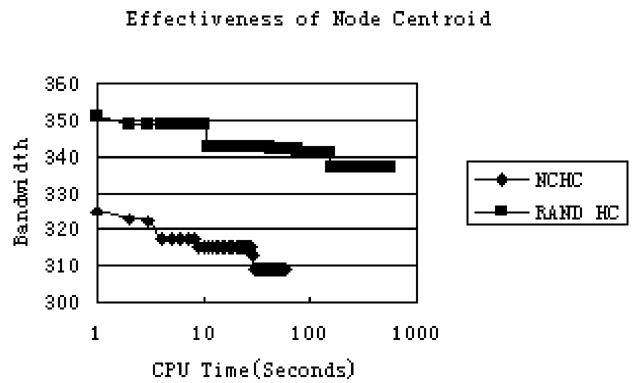
Table 4: Results on DWT data set of Harwell-Boeing Sparse Matrix Collection

Size	GPS	MATLAB	WBRA	TS (Esposito)	NCHC	FNCHC
59	8	8	7	9	**6**	**6**
66	**3**	**3**	**3**	**3**	**3**	**3**
72	8	7	7	7	**5**	6
87	21	18	13	14	**10**	**10**
162	22	16	**13**	15	**13**	**13**
193	64	57	35	57	**32**	34
209	33	33	30	38	**23**	25
245	83	55	27	25	**23**	24
307	47	44	32	36	**31**	32
361	15	15	**14**	**14**	16	17
503	78	59	51	76	**43**	48
592	121	42	39	88	**31**	33
869	97	43	39	58	**34**	36
1005	196	65	68	92	**61**	64
$B_f(G)$	56.86	33.21	27.00	38.00	23.64	25.07
Best	1	1	3	2	13	4

We can find in table 4 that the NCHC and FNCHC achieve the best results. The average running time for our FNCHC is 0.41 s. Since the other experiments have been run on the IBM RS6000 250T, we cannot tell the CPU speed and thus have not compared running time. We conclude that experiments on the three test sets show that our new NCHC method achieves the best solution in quality on the bandwidth minimization problem compared with the other well-known algorithms, including the most-recently developed GRASP_PR. Further, the fast version, FNCHC, obtains high quality solutions in short times.

5.2 Effectiveness of Node Centroid

Experimentation results show that our NCHC can achieve best solutions in much faster speed than recently developed heuristic algorithms. To analyze the effectiveness of our Node Centroid Method in acting as the globe search, we have compared our NCHC with the RAND_HC which perform Hill Climbing with random start sequences. We compare the two procedure on the bp_1000 instance from the second test set we used. The comparison result is show in the following figure.



In the comparison, we let both procedures run for Hill Climbing 7500 times. From figure 5.2 we can find that the NCHC get much better result than the RAND_HC. And the NCHC finish all the 7500 times Hill Climbing in 59 s, while it cost 556 s for the RAND_HC to finish 7500 times Hill Climbing. To explain why our Node Centroid Method has increased the speed for the whole procedure significantly, we have recorded the average swap times in the Hill Climbing Procedure for the NCHC and RAND_HC. In th e NCHC the average swap times is 123, while the average swap times in the RAND_HC is 2036. This result show that our new Node Centroid Method has explored good solution regions effectively. Therefore only a few steps of swaps needed to improve the soluti on quality in the Hill Climbing part.

6 Conclusions

We have proposed a Node Centroid adjustment method with Hill Climbing for the well-known matrix bandwidth minimization problem where node adjustments are intrinsically a natural strategy in reducing graph node labelings contributing to bandwidth. Further, we have catered for a range of vertices that contribute to bandwidth by using an adjustable parameter to include these. Experimentation has shown that the Node Centroid global search works well with Hill Climbing for this problem. Best solutions in quality are achieved by the new NCHC algorithm, while the FNCHC provides good solution quality at fast speeds and is comparable in speed to the fast GPS algorithm.Experimentation results also show that our new Node Centroid method has explore good solution regions effectively, which indicates that we can apply the new Node Centroid procedu re to other similar combinatorial optimization problems, such as matrix profile reduction and Minimum Linear Arrangement Problem.

Acknowledgments

The authors would like to thank Professors Rafael Marti and Vicente Campos for providing them with their executable files and the description of the their test data.

References

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

[2] Berry, M., Hendrickson, B., Raghavan,P.,1996. Sparse matrix reordering schemes 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, 1996.

[3] Blum, A., Konjevod, G., Ravi, R., and Vempala, S. 1998, Semidefinite relaxations for minimum bandwidth and other vertex-ordering problems,Proceedings of the 30th Annual Symposium on the Theory of Computing, 1998.

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

[5] Diaz, J., Petit, J., and Serna M., A survey on graph layout problems, 2002, ACM Computing Surveys, 34(3):313-356, 2002.

[6] Feige U., Approximating the Bandwidth via Volume Respecting Embeddings, 1998, Proceedings of the 30th Annual Symposium on the Theory of Computing, 1998.

[7] Garey, M., Graham, R., Johnson, D., Knuth, D., 1978. Complexity results for bandwidth minimization. SIAM J. Applied Mathematics,pp. 34:477-495, 1978.

[8] 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), pp. 236-250.

[9] Esposito, A., Catalano, M., S., Malucelli F., Tarricone, L. 1998. Sparse Matrix Bandwidth Reduction: Algorithms, applications and real industrial cases in electromagnetics, 1998. High Performance Algorithms for Structured Matrix Problems, Advances in the theory of Computation and Computational Mathematics Volume 2, pp. 27-45.

[10] Esposito, A., Catalano, M., S., Malucelli F., Tarricone, L. 1998. A new matrix bandwidth reduction algorithm, 1998. Operation Research Letters 23, pp. 99-107.

[11] Lim, A., Rodrigues, B., Xiao, F.,An Evolutionary Approach to Bandwidth Minimization, 2003, Congress on Evolutionary Computation, 2003.

[12] Lim, A., Rodrigues, B., Xiao, F., A New Heuristic Method for the Bandwidth Minimization Problem,2003, Fifth International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 2003 .

[13] Lim, A., Lin, J., Xiao, F., Particle Swarm Optimization and Hill Climbing to Solve the Bandwidth Minimization Problem, 2003, The Fifth Metaheuristics International Conference.

[14] Linial, N., London, E., and Rabinovich Yu., 1995, The geometry of graphs and some of its algorithmic applications, Combinatorica, 15, 215 - 245, 1995.

[15] 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.

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

[17] Pinana, E., Plana, I., Campos, V. Marti, R., 2002, GRASP and Path Relinking for the Matrix Bandwidth Minimization. In print, European Journal of Operational Research. http://www.uv.es/ rmarti/.