A New Branch and Bound Algorithm for the Cyclic Bandwidth Problem

Hillel Romero-Monsivais, Eduardo Rodriguez-Tello, and Gabriel Ramírez

CINVESTAV-Tamaulipas, Information Technology Laboratory.
Km. 5.5 Carretera Victoria-Soto La Marina, 87130 Victoria Tamps., Mexico
{hromero,ertello,grtorres}@tamps.cinvestav.mx

Abstract. The *Cyclic Bandwidth* problem (CB) for graphs consists in labeling the vertices of a guest graph G by distinct vertices of a host cycle C (both of order n) in such a way that the maximum distance in the cycle between adjacent vertices in G is minimized. The CB problem arises in application areas like VLSI designs, data structure representations and interconnection networks for parallel computer systems.
In this paper a new *Branch and Bound* (B&B) algorithm for the CB problem is introduced. Its key components were carefully devised after an in-depth analysis of the given problem. The practical effectiveness of this algorithm is shown through extensive experimentation over 20 standard graphs. The results show that the proposed exact algorithm attains the lower bounds for these graphs (of order $n \leq 40$) expending a reasonable computational time.

Key words: Branch and Bound, Cyclic Bandwidth Problem

1 Introduction

The *Cyclic Bandwidth* problem (CB) was first stated by Leung *et al.* in 1984 [1] in relation with the design of a ring interconnection network for a set of computers (V) whose communication pattern was described by a graph $G(V, E)$ where $\{u, v\} \in E$ if computer u communicates with computer v. They were interested in finding an arrangement of these computers on a cycle so that every message sent can arrive at its destination in at most k steps. The CB problem arises also in other application areas like VLSI designs [2], data structure representations [3], code design [4] and interconnection networks for parallel computer systems [5].

The CB problem can be formally defined as follows. Let $G(V, E)$ be a finite undirected graph, where V ($|V| = n$) defines the set of vertices and $E \subseteq V \times V = \{\{u, v\} \mid u, v \in V\}$ is the set of edges. Given a bijective labeling function for the vertices of G, $\varphi : V \rightarrow \{1, 2, \ldots, n\}$, the cyclic bandwidth (the cost) for G with respect to the labeling φ is defined as:

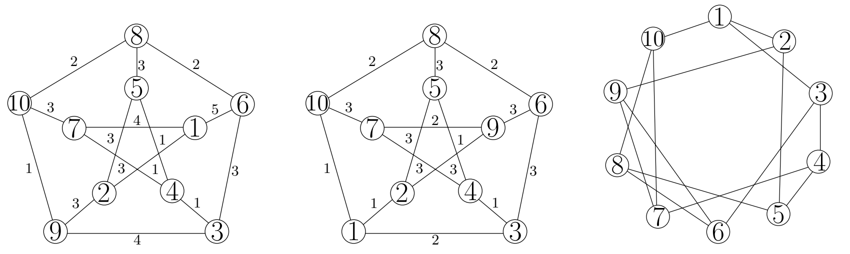
$$B_c(G, \varphi) = \max_{(u,v) \in E} \{|\varphi(u) - \varphi(v)|_n\}, \qquad (1)$$

where $|x|_n = \min\{|x|, n - |x|\}$ for $0 < |x| < n$, which is called *cyclic distance*. Then the CB problem consists in finding a labeling φ^*, which minimizes $B_c(G, \varphi^*)$, i.e.,

$$B_c(G, \varphi^*) = \min\{B_c(G, \varphi) : \varphi \in \mathscr{L}\}\,, \tag{2}$$

where $\mathscr{L}$ is the set of all possible labelings.

For instance, consider the graph $G(V, E)$ depicted in Fig. 1(a), consisting of ten vertices with a labeling φ given by the numbers shown inside each vertex. The cyclic distance between each pair of adjacent vertices $(u, v) \in E$ can be calculated using the expression $\min\{|\varphi(u) - \varphi(v)|, n - |\varphi(u) - \varphi(v)|\}$. These cyclic distances are represented by the numbers associated to the edges in the graph. For this particular labeling φ, $B_c(G, \varphi) = 4$. On the other hand, if the labels 1 and 9 are exchanged a new labeling φ' is obtained (see Fig. 1(b)), which produces a smaller (better) cyclic bandwidth value ($B_c(G, \varphi') = 3$). The embedding of this new labeling of the graph G in a cycle is presented in Fig. 1(c) for illustrative purposes.



(a) Labeling φ **(b)** Labeling φ' **(c)** Embedding G in a cycle

Fig. 1. Example of a *Cyclic Bandwidth* problem instance.

It was demonstrated that finding the cyclic bandwidth is NP-complete for general graphs [6]. It is important to note that the CB problem is a natural extension of the well-known *bandwidth minimization* problem for graphs [7], which consists in embedding the vertices of a guest graph G in a host path P (both of order n) in such a way that the maximum distance in the path between adjacent vertices in G is minimized.

In this paper, a new *Branch and Bound* (B&B) algorithm for finding exact solutions for the CB problem for general graphs is presented. The proposed B&B algorithm is based on a lowest-cost search strategy, i.e., branches having the lowest cost are explored while branches with higher cyclic bandwidth are cut. It implements the Hungarian method [8] for solving the problem of efficiently computing the cost of assigning every available label in a partial solution to each possible non-labeled vertex. The practical usefulness of the proposed B&B is assessed experimentally using a test-suite, composed by 20 standard graphs belonging to 4 different classes. The computational results from this experiments show that our algorithm is able to attain the lower bounds for these graphs (of order $n \leq 40$) expending a reasonable computation time.

The remainder of this paper is organized as follows. In Sect. 2 a brief review is given to present some representative related work. The main components of the proposed B&B algorithm are presented in detail in Sect. 3. Section 4 describes the experimental methodology used for assessing the practical usefulness of the proposed B&B algorithm, while the computational results produced by these experiments are presented in Sect. 5. Finally, the conclusions of this work and some possible directions for future research are provided in Sect. 6.

2 Relevant Related Work

In spite of its practical and theoretical importance, less attention has been paid to the CB problem with respect to other graph labeling problems. Up to now, most of the research on this important labeling problem has been devoted to the theoretical study of its properties with the aim of finding exact solutions for certain specific cases. Next, we present a brief review of these studies.

In 2002 Zhou [9] proposed a systematic method for obtaining lower bounds for the bandwidth and cyclic bandwidth problems in terms of some distance- and degree-related parameters of the graph. The main idea of this method is to relax the condition of embedding the graph G on the host graph with the aid of a graphical parameter possessing some kind of monotonic property. This method has been demonstrated to be efficient when the parameters are chosen appropriately. The author concludes that this method yields a number of lower bounds for the ordinary and cyclic bandwidths. In both cases, it gives rise to new estimations, as well as improvements of some known results.

Later, de Klerk *et al.* [10] proposed two new semidefinite programming (SDP) relaxations of the bandwidth and cyclic bandwidth based on the quadratic assignment problem (QAP) reformulation. The bounds produced by this method were tested for some special graphs showing that they are tight for paths, cliques, complete bipartite graphs. However, these bounds are not tight for hypercubes, rectangular grids and complete k-level t-ary trees.

In 1995 Yuan and Zhou [11] demonstrated that for unit interval graphs, there exists a labeling which is simultaneously optimal for the following seven labeling problems: bandwidth, cyclic bandwidth, profile, cutwidth, modified cutwidth and bandwidth sum. Following this idea, in [12] Lam *et al.* made a characterization of graphs with equal bandwidth and cyclic bandwidth which includes planar graphs, triangulation meshes and grids with some specific characteristics.

To the best of our knowledge, there has been no previous research investigating *Branch and Bound* based algorithms for solving the CB problem in the case of general graphs. Therefore, this paper presents an original contribution in this field.

3 A New Branch and Bound Algorithm

In this section, we present a new exact algorithm, based on *Branch and Bound* (B&B), for solving the CB problem. This algorithm employs a lowest-cost search strategy, where branches having the lowest cost are explored while branches with higher cost are cut. The following subsections describe in detail the main components of our algorithm.

3.1 Branching Strategy

Every possible labeling φ_k, for $k = \{1, 2, 3, ..., (n-1)!/2\}$, must be tested. In our algorithm, the labelings are tested in lexicographic order, beginning with $\varphi_1 = \{1, 2, 3, ..., n\}$ and ending with $\varphi_{(n-1)!/2} = \{n, n-1, n-2, ..., 1\}$. Each individual label in φ_k is assigned to vertices, one at a time, producing three different sets: V' the set of already labeled vertices, V''_k the set of unlabeled vertices which are adjacent to those in V' and φ''_k the lexicographic order set of available labels.

The cyclic bandwidth (cost) $B_c(G, \varphi'_k)$ for every particular partial labeling is computed considering V' in G. If all labels in φ_k have been assigned, then a labeling φ'_k with a value $B_c(G, \varphi'_k)$ smaller than the prefixed bound b has been found. The algorithm then stores this new best solution in φ^σ.

According to Leung [1], the upper bound for the CB is $n/2$, so at the beginning $b = n/2$. When a labeling φ^σ is found, the search retakes a new bound $b = B_c(G, \varphi^\sigma) - 1$.

The algorithm performs a depth-first search, so if a certain label is assigned to a vertex and $B_c(G, \varphi'_k) > b$, the branch is cut. Then, a backtrack to the parent label is applied and then the algorithm advances to the next child label. The search process stops when all labelings have been analyzed. A detailed description of the proposed B&B method is summarized in Algorithm 1.

3.2 Assignment Strategy

Two important issues are considered when adding a new label to a partial solution φ'_k: the cost generated by that new label, and the lowest cost which can be produced by assigning the available labels φ''_k to the set V''_k of unlabeled vertices adjacent to those in V'. The latter provides information allowing to know if the adjacent vertices can be labeled without exceeding the current bound b in further iterations.

The Hungarian method is an algorithm which solves the assignment problem in polynomial time [8]. Our implementation employs this algorithm to compute the cost of assigning each label $l \in \varphi''_k$ to each possible vertex $v \in V''_k$. However, if the cost of an assignment becomes higher than the current bound b, then the evaluation of φ_k is not completed.

For instance, consider the partial representation of a graph $G = (V, E)$ depicted in Fig. 2, which consists of five vertices labeled with the number shown within each vertex. Let $n = 10$ and $b = 3$ for this example. Since the vertices in

Algorithm 1: B&B for the CB problem

```
B&B($G(E,V)$)
$b = \frac{n}{2}$
$i \leftarrow 1$
$\varphi_i = \{1,2,3,...,n\}$
$V' \leftarrow \varphi' \leftarrow \varphi^\sigma \leftarrow \emptyset$
while $i \leq (n-1)!/2$ do
    $V_i \leftarrow \varphi_i$
    $\varphi' \leftarrow \varphi' \cup \{\varphi_i\}$
    $V' \leftarrow V' \cup \{V_i\}$
    $c \leftarrow B_c(G,\varphi')$
    if $c > b$ then
        $j \leftarrow$ backtracking $(\varphi', i)$
        if $j > 0$ then
            $i \leftarrow j$
            $V_i \leftarrow \varphi'_i$
            $V' \leftarrow V' \cup \{V_i\}$
        else return $\varphi^\sigma$
    $V'' \leftarrow \{v \in (V \backslash V') | (v,u) \in E, u \in V'\}$
    $\varphi'' \leftarrow \{l \in (\varphi \backslash \varphi')\}$
    assignment $(V'', \varphi'')$
    $c \leftarrow B_c(G,\varphi')$
    if $c > b$ then
        $j \leftarrow$ backtracking $(\varphi', i)$
        if $j > 0$ then
            $i \leftarrow j$
            $V_i \leftarrow \varphi'_i$
            $V' \leftarrow V' \cup \{V_i\}$
        else return $\varphi^\sigma$
    if $i = n$ then
        $\varphi^\sigma \leftarrow \varphi'$
        $b \leftarrow b - 1$
        $j \leftarrow$ backtracking $(\varphi', i)$
        if $j > 0$ then
            $i \leftarrow j$
            $V_i \leftarrow \varphi'_i$
            $V' \leftarrow V' \cup \{V_i\}$
        else return $\varphi^\sigma$
    $i \leftarrow i + 1$
```

the set V are taken following a lexicographic order, then the last labeled vertex was $V_6 \leftarrow 5$, therefore $V' = \{...V_5, V_6\}$, $\varphi'_k = \{..., 3, 5\}$ and $V''_k = \{V_7, V_8, V_9\}$. Suppose that $\varphi''_k = \{1, 4, 9, 10\}$ and for this partial labeling $B_c(G, \varphi'_k) < b$. Using the Hungarian method the costs of assigning the labels in φ''_k to the vertices

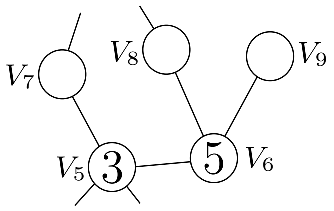


Fig. 2. Example of a partially labeled graph.

in V''_k are estimated and presented in Table 1. From this table it is clear that V_7 and V_8 could be labeled without exceeding the current bound b in further iterations. However, $B_c(G, \varphi'_k) \leq b$ could not be guaranteed for V_9. In this case, the evaluation of φ_k is stopped making a backtracking which produces the next possible labeling φ_{k+1} in lexicographic order.

Table 1. Table assignment vertex-label

	φ''_k			
V''_k	*1*	*4*	*9*	*10*
v_7	**2**	1	5	3
v_8	4	**1**	4	5
v_9	4	1	4	5

After preliminary experimentation with this *basic assignment strategy* we have observed that it could be improved. The amelioration consists in also analyzing the assignments of the adjacencies between the vertices belonging to the set V''_k. Although it represents an additional computational effort, it permits to find at this step vertices producing a cost which exceeds the current bound b. The main advantage of this *improved assignment strategy* is that it is able in certain cases to stop the evaluation of φ_k earlier than the basic assignment strategy, which reduces importantly the size of the search space.

3.3 Backtracking Strategy

A backtracking occurs if the cyclic bandwidth of the partial labeling φ'_k exceeds the current bound b when its i-th label is added. This backtracking procedure is performed as described in Algorithm 2. First, the best element in the set φ''_k of available labels is identified. It is the first label $l \in \varphi''_k$ which guarantees that the resulting cyclic bandwidth $B_c(G, \varphi'_k)$ is lower than the current bound b when this label is assigned to the i-th position of the partial labeling φ'_k. If there is no such label meeting this condition, the $(i-1)$-th position of φ'_k is analyzed (a backtracking occurs), the set φ''_k is updated and its best label is searched again.

This process is repeated until the best label is found or the complete set φ'_k was analyzed.

Algorithm 2: Backtracking strategy

```
1  backtracking ($\varphi'$, $i$)
2  $\varphi''$ Available labels
3  while true do
4      $\varphi'_i \leftarrow$ best($\varphi''$)
5      if $\varphi'_i = \emptyset$ then
6          $i \leftarrow i - 1$
7          if $i > 0$ then
8              $\varphi'' \leftarrow \varphi'' \cup \{\varphi'_i\}$
9              $\varphi'_i \leftarrow \emptyset$
10         else return $i$
11     else return $i$
```

4 Experimental Setup

In order to assess the performance of the proposed B&B algorithm introduced in Sect. 3 it was was coded in C and compiled with *gcc* using the optimization flag $-O3$. It was run sequentially into a cluster composed of 4 processors Xeon X5650 equipped with six cores at 2.66 GHz, 32 GB of RAM and Linux operating system.

Due to the deterministic nature of the algorithm, it was executed only one time over each of the selected benchmark instances. The maximum CPU time allowed for each execution was predefined to 36 hours.

4.1 Compared Algorithms

For the computational experiments two different versions of the B&B algorithm described in Sect. 3 were implemented. The main difference between these two versions is the assignment strategy employed. The first one, denoted B&B1, employs the *basic assignment strategy*, which employs the Hungarian method to solve the problem of finding the cost (cyclic bandwidth) of assigning each label $l \in \varphi''_k$ to each possible vertex $v \in V''_k$. The second version, called B&B2, uses the *improved assignment strategy* which also analyzes the assignments of the adjacencies between the vertices belonging to the set V''_k (see Sect. 3.2).

4.2 Benchmark Instances and Performance Assessment

As it was mentioned at the end of Sect. 2, there has been no previous research investigating neither exact nor approximate algorithms for solving the CB prob-

Table 2. Characteristics of the test-suite used in the experiments. It consists of 20 instances representing 4 different classes of graphs.

Graph	$\lvert V\rvert = n$	$\lvert E\rvert = m$	$ED = 2m/n(n-1)$	$\mathcal{T}$
cycle20	20	20	0.105	1
cycle25	25	25	0.083	1
cycle30	30	30	0.069	1
cycle35	35	35	0.059	1
cycle40	40	40	0.051	1
grid5x4	20	31	0.163	4
grid5x5	25	40	0.133	5
grid5x6	30	49	0.113	5
grid5x7	35	58	0.097	5
grid5x8	40	67	0.086	5
path20	20	19	0.100	1
path25	25	24	0.080	1
path30	30	29	0.067	1
path35	35	34	0.057	1
path40	40	39	0.050	1
tree21	21	20	0.095	3
tree25	25	24	0.080	4
tree31	31	30	0.065	4
tree33	33	32	0.061	4
tree35	35	34	0.057	4

lem in the case of general graphs. There exists no standard test-suite reported in the literature for evaluating the performance of this kind of algorithms. For this reason a test-suite composed of 20 benchmark instances is proposed in this paper.

The test-suite comprises 4 different classes of standard graphs: cycles, grids, paths and perfect binary trees. All these instances have a number of vertices between 20 and 40 and are publicly available[1]. The characteristics of these instances are detailed in Table 2, where the first three columns indicate the name of the graph as well as its number of vertices and edges. The edge density $(2|E|/|V|(|V|-1))$ of each graph is depicted in Column 4, while the last column presents the theoretical lower bounds $\mathcal{T}$ reported in [10, 13, 12].

The criteria used for evaluating the performance of the compared B&B algorithms are the same as those used in the literature [14, 15]: the total number of partial solutions (φ'_k) examined during the search process, the total number of backtracking operations occurred and the expended CPU time in seconds. For these criteria smaller values are better.

[1] `http://www.tamps.cinvestav.mx/~ertello/cbmp.php`

5 Computational Results

The experimental comparison between the two versions of the *Branch and Bound* algorithm introduced in this paper (B&B1 and B&B2) was performed over the benchmark instances described above and employing the same computational platform introduced in Sect. 4.

The results from this experiment are summarized in Table 3, which lists in the first column the name of the benchmark instance. For each compared algorithm this table presents: the total number of partial solutions φ'_k evaluated during the search process ($\#\varphi'_k$), the total number of backtracking operations occurred (B), and the CPU time in seconds (*time*) expended for finding the optimal solution $\varphi*$. Instances marked with the symbol "—" indicate that they could not be solved within the maximum allowed time of 36 hours. Given that we are comparing exact algorithms, it is not necessary to report the cyclic bandwidth achieved as it is always the optimum value.

Table 3. Performance comparison between two different versions (B&B1 and B&B2) of the *Branch and Bound* algorithm proposed in Sect. 3.

	B&B1			B&B2		
Graph	$\#\varphi'$	B	*time*	$\#\varphi'$	B	*time*
cycle20	2331	2042	0.01	586	487	0.01
cycle25	12900	11867	0.12	627	514	0.01
cycle30	88516701	81990664	1007.90	2223754	2089272	43.10
cycle35	465326343	431788129	4184.05	46183164	43443834	449.58
cycle40	397409592	372614337	4996.08	23409592	22614337	531.08
grid5x4	123230	113472	0.82	18280	17002	0.16
grid5x5	2674635	2517186	29.32	562911	532316	8.43
grid5x6	144312826	135950146	3016.46	3253192	3099321	84.63
grid5x7	—	—	—	18490735	17431435	426.32
grid5x8	—	—	—	160963339	155466427	14144.13
path20	695	585	0.01	559	457	0.01
path25	19997	18023	0.07	2498	2227	0.01
path30	101821	93156	7.63	2847814	2513041	0.60
path35	74072	70352	0.83	10393	9815	0.13
path40	—	—	—	10045277883	9116835089	34023.25
tree21	147232	133651	0.47	61483	56644	0.23
tree25	223355577	204264652	763.27	28450274	26320726	110.92
tree31	707968183	663767447	3552.96	220801934	208577566	1240.82
tree33	2248823425	2121853157	10965.57	882035992	836064022	4588.00
tree35	—	—	—	16419099889	15388330578	11288.81

Analyzing the data presented in Table 3 lead us to the following main observations. First, the most time-consuming algorithm is B&B1, despite B&B2

employs the improved assignment strategy which demands an additional computational effort. This is possible because the improved assignment strategy used by B&B2 is able to importantly reduce the size of the explored search space (see column $\#\varphi'_k$).

Second, one observes that B&B1 was not able to find a solution within the time limit of 36 hours for the following classes of graphs with $n \geq 35$: grids, paths and perfect binary trees. On the contrary, B&B2 found an exact solution for all the selected instances employing at most 9.45 hours and consistently a lower number of backtrack operations (B) than B&B1.

The comparison of the computational time expended by B&B1 and B&B2 is better illustrated in Fig. 3. The plot represents the studied instances (ordinate) against the CPU time in seconds expended by the two compared algorithms using a $\log_{10}$ scale (abscissa). From this graph one clearly observes that B&B2 consistently expends much less CPU time than B&B1.

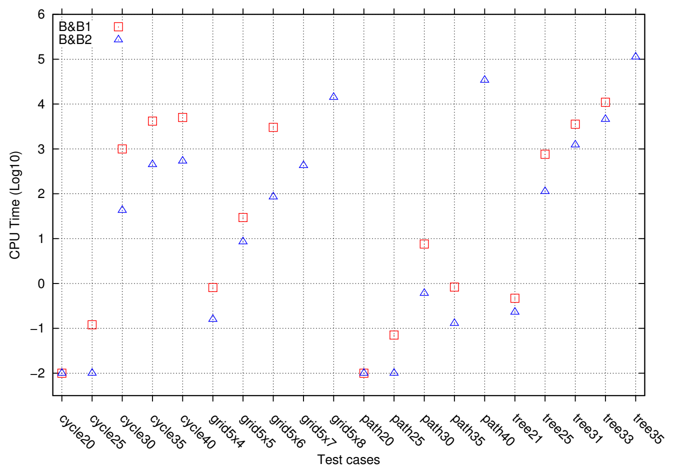


Fig. 3. Consumed CPU time comparison between two different versions (B&B1 and B&B2) of the proposed *Branch and Bound* algorithm.

6 Conclusions and Further Work

In this paper a new *Branch and Bound* (B&B) algorithm for the Cyclic Bandwidth (CB) problem was introduced. It is based on a lowest-cost search strategy which explores the branches having the lowest cost while it cuts the branches with higher cyclic bandwidth.

Two different versions of this B&B algorithm, employing slightly different assignment strategies, were implemented. The main difference between these two versions is that the first one, denoted B&B1, employs a *basic assignment* *strategy* based on the Hungarian method [8] for efficiently finding the cost of assigning every available label in a partial solution to each possible non-labeled vertex; while the second, called B&B2, uses an *improved assignment strategy* which additionally analyzes the assignments of the vertices belonging to the set of unlabeled vertices adjacent to those in the set of already labeled vertices.

Given the lack in the literature of a standard test-suite for evaluating the performance of algorithms for solving the CB problem, a new benchmark composed of 20 instances, comprising 4 different classes of standard graphs (cycles, grids, paths and perfect binary trees), was proposed in this work.

A performance experimental comparison between the two versions of the *Branch and Bound* algorithm introduced in this paper (B&B1 and B&B2) was performed over the benchmark instances proposed. The results from this comparison show that B&B2 consumes much less CPU time (up to 97% in some cases), because it analyzes fewer partial solutions before finding the lower bound of the instances. This gives us the opportunity to find exact solutions for graphs than B&B1 was unable to solve.

This work opens up a range of possibilities for future research. Currently, we are interested in hybridizing our B&B algorithm with some kind of metaheuristic in order to speed the convergence time of the resulting algorithm, inspired in the ideas presented in [15]. The parallelization of the B&B algorithm introduced in this paper also represents an interesting issue for future work.

Acknowledgments

This research work was partially funded by the following projects: CONACyT 99276, Algoritmos para la Canonización de Covering Arrays; 51623 Fondo Mixto CONACyT y Gobierno del Estado de Tamaulipas.

References

1. Leung, J., Vornberger, O., Witthoff, J.: On some variants of the bandwidth minimization problem. SIAM Journal on Computing **13**(3) (1984) 650–667
2. Bhatt, S.N., Thomson Leighton, F.: A framework for solving VLSI graph layout problems. Journal of Computer and System Sciences **28**(2) (1984) 300–343
3. Rosenberg, A.L., Snyder, L.: Bounds on the costs of data encodings. Theory of Computing Systems **12**(1) (1978) 9–39
4. Chung, F.R.K.: Labelings of graphs. In Beineke, L.W., Wilson, R.J., eds.: Selected topics in graph theory volume 3. Academic Press (1988) 151–168
5. Hromkovič, J., Müller, V., Sýkora, O., Vrťo, I.: On embedding interconnection networks into rings of processors. Lecture Notes in Computer Science **605** (1992) 51–62
6. Papadimitriou, C.H.: The NP-completeness of the bandwidth minimization problem. Computing **16**(3) (1976) 263–270
7. Harper, L.: Optimal assignment of numbers to vertices. Journal of SIAM **12**(1) (1964) 131–135
8. Kuhn, H.W.: The Hungarian method for the assignment problem. Naval Research Logistic Quarterly **2**(1) (1955) 83–97

9. Zhou, S.: Bounding the bandwidths for graphs. Theoretical computer science **249**(2) (2002) 357–368
10. de Klerk, E., Eisenberg-Nagy, M., Sotirov, R.: On semidefinite programming bounds for graph bandwidth. Technical report, Centrum Wiskunde & Informatica, (2011)
11. Yuan, J., Zhou, S.: Optimal labelling of unit interval graphs. Applied Mathematics, A Journal of Chinese Universities **10B**(3) (1995) 337–344
12. Lam, P.C.B., Shiu, W.C., Chan, W.H.: Characterization of graphs with equal bandwidth and cyclic bandwidth. Discrete Mathematics **242**(3) (2002) 283–289
13. Lam, P.C.B., Shiu, W.C., Chan, W.H.: On bandwidth and cyclic bandwidth of graphs. Ars Combinatoria **47**(3) (1997) 147–152
14. Martí, R., Campos, V., Piñana, E.: A branch and bound algorithm for the matrix bandwidth minimization. European Journal of Operational Research **186**(2) (2008) 513–528
15. Palubeckis, G., Rubliauskas, D.: A branch-and-bound algorithm for the minimum cut linear arrangement problem. Journal of Combinatorial Optimization (2011) 1–24, DOI: 10.1007/s10878–011–9406–2