An Approximation Algorithm for the Bandwidth Problem on Dense Graphs

Marek Karpinski* **Jürgen Wirtgen†**
Alex Zelikovsky‡

TR-97-016

May 1997

Abstract

The *bandwidth problem* is the problem of numbering the vertices of a given graph G such that the maximum difference between the numbers of adjacent vertices is *minimal.* The problem has a long history and is known to be NP-complete [Papadimitriou, 1976]. Only few special cases of this problem are known to be efficiently approximable. In this paper we present the first constant approximation ratio algorithms on dense instances of this problem.

*Dept. of Computer Science, University of Bonn, 53117 Bonn, and International Computer Science Institute, Berkeley, California. Research partially supported by DFG Grant KA 673/4-1, by the ESPRIT BR Grants 7097 and EC-US 030, and by the Max-Planck Research Prize. Email: marek@cs.bonn.edu.

†Dept. of Computer Science, University of Bonn, 53117 Bonn. Research partially supported by the ESPRIT BR Grants 7097 and EC-US 030. Email: wirtgen@cs.bonn.edu

‡Dept. of Computer Science, University of Bonn, 53117 Bonn. Visiting from Dept. of Computer Science, Thornton Hall, University of Virginia, VA 22903. Research partially supported by Volkswagen Stiftung and Packard Foundation. Email: alexz@cs.virginia.edu

1 Introduction

The bandwidth problem on graphs has a very long and interesting history cf. [CCDG 82].

It originated around 1962 at the Jet Propulsion Laboratory (JPL) at Pasadena as a model for minimizing absolute and average errors of the 6-bit picture codes on a hypercube. The bandwidth problem for matrices seems to originate even earlier in the 1950s as a model for manipulating structural hardware matrices

Despite its very long history and its technical importance, it is not much known about efficient approximability of the bandwidth problem.

Formally the bandwidth minimization problem is defined as follows. Let $G = (V, E)$ be a simple graph on n vertices. A numbering of G is a one-to-one mapping $f : V \to \{1, ..., n\}$. The bandwidth $B(f, G)$ of this numbering is defined by

$$B(f, G) = \max\{|f(v) - f(w)| : \{v, w\} \in E\},$$

the greatest distance between adjacent vertices in G corresponding to f. The bandwidth $B(G)$ is then

$$B(G) = \min_{f \text{is a numbering of } G} \{B(f, G)\}$$

Clearly the bandwidth of G is the greatest bandwidth of its components.

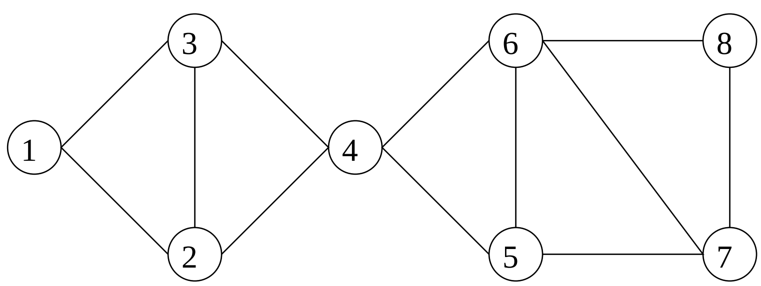


Figure 1: A small 1/4-dense graph G. It has 8 vertices and minimum degree 2.

The problem of finding the bandwidth of a graph is NP-complete [Pa 76], even for trees with maximum degree 3 [GGJK 78]. The general problem is not known to have any sublinear n^ϵ-approximation algorithms. There are only few cases where we can find the optimal layout in polynomial time. Saxe [Sa 80] designed an algorithm which decides whether a given graph has bandwidth at most k in time $O(n^k)$ by dynamic programming. Bandwidth two can be checked in linear time [GGJK 78]. Kratsch [Kr 87] introduced an exact $O(n^2 \log n)$ algorithm for the bandwidth problem in interval graphs. Smithline [Sm 95] proved that the bandwidth of the complete k-ary tree $T_{k,d}$ with d levels and k^d leaves is exactly $\lceil k(k^d-1)/(k-1)(2d)\rceil$. Her proof is constructive and contains a polynomial time algorithm, which do this task. For catapillars [HMM 91] found a polynomial time $\log n$-approximation algorithm. A catapillar is a special kind of a tree consisting of a simple chain, the body, with an arbitrary number of simple chains, the hairs, attached to the body by coalescing an endpoint of the added chain with a vertex of the body. Although they are almost interval graphs, the bandwidth problem restricted to catapillars is NP-complete.

In this paper we present the first constant approximation ratio algorithm for δ-dense graphs. In particular we construct a 3-approximation algorithm. We call a graph G δ-dense, if the minimum degree $\delta(G)$ is at least δn (see e.g. [AKK 95]). To introduce our method, we describe in Section 2 a weaker version of the algorithm - namely a 4-approximation algorithm. It uses as one of its building-blocks the construction of perfect matchings in bipartite graphs. Furthermore it is easy to parallelize, since the perfect matching problem lies in RNC. Recently there has been some success in designing parallel approximation algorithms for some other hard problems [PS 97] [Tr 97] [TX 97].

This paper is organized as follows. In Section 2 we outline a 4-approximation algorithm. Section 3 gives a refinement to a 3-approximation algorithm and section 4 gives a 2-approximation algorithm for dense directed graphs.

2 Outline of the 4-Approximation Algorithm

Suppose we have some optimal numbering. Then we can split this layout in $n/B(G)$ boxes, so that there are only edges between neighbored boxes (see figure 2). It is clear that a graph with minimum degree k has at least bandwidth k. Therefore the bandwidth of δ-dense graphs is at least δn and thus we have at most $1/\delta \in O(1)$ boxes. Without loss of generality we may assume that n is divedable by $B(G)$, else we can construct a new graph G' with the same bandwidth, by adding a chain of clique of size $\delta/2n$ to G until $|V(G')|$ is dividable by $B(G)$.

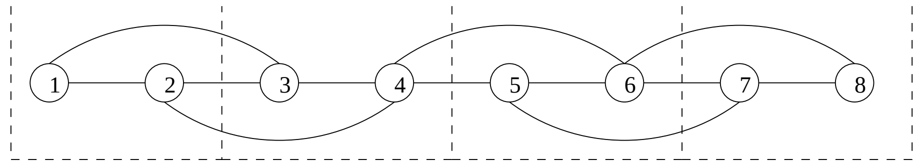


Figure 2: An optimum layout of the graph G in figure 1. It is optimum, because $\delta(G) = 2$ and the maximum distance of two neighbored vertices is 2.

By repeating the algorithm for all the possible values for the bandwidth, we can get for certain the right value. Note, that there are only $O(n)$ possible values. The algorithm chooses at random $O(\log n)$ vertices $R \subseteq V$. For a vertex $v \in V \setminus R$ we call the neighbors in $N(v) \cap R$ the roots of v. We have 2 important properties of R:

1. R forms with high probability a dominating set (Lemma 1)
2. With high probability each of the boxes has at least one representative in R (Lemma 2).

Lemma 1 *Let $G = (V, E)$ be a δ-dense graph. A set of*

$$k = \frac{\log(n/\alpha)}{\log(1/(1-\delta))} = O(\log n)$$

randomly chosen vertices R forms with probability at least $(1-\alpha)$ a dominating set.

PROOF: The probability, that one particular vertex v will be dominated by one randomly chosen vertex, is at least δ. If we choose k vertices independently, then the probability that it is not dominated, is at most $(1-\delta)^k$. Thus the expected number of not dominated vertices is at most α, because

$$\begin{aligned} (1-\delta)^k n &\leq \alpha \\ n/\alpha &\leq (1/(1-\delta))^k \\ \frac{\log(n/\alpha)}{\log(1/(1-\delta))} &\leq k \end{aligned}$$

By Markov's inequality we get the lemma. ■

Lemma 2 *Let V be a finite set and $V = V_1 \dot\cup V_2 \dot\cup \ldots \dot\cup V_c$ with $|V_i| = \lceil n/c \rceil$. Choose independently $k \log n$ vertices $v \in_R V$ at random, forming a set R. Then we have with high probability for each V_i a representative in R.*

PROOF: The probability that there is no representative in R for a particular V_i, is $\left(\frac{c-1}{c}\right)^{k \log n}$. Thus the expected number of V_i which do not have any representative in R is $c\left(\frac{c-1}{c}\right)^{k \log n}$. By Markov's inequality we get, that the probability, that a V_i has not a representative is at most $2cn^{k \log((c-1)/c)} = o(1)$. ■

Suppose we know to which box each root belongs to. In fact we can find the right assignment of the roots to the boxes by exhaustive search in polynomial time. Observe that we have only a constant number of boxes and that the size of R is in the order of $\log n$. So there are only $O(1)^{O(\log n)} = n^{O(1)}$ possibilities. For any vertex which is not a root we have now at most 3 possible boxes where it belongs to, because it has at least one root (Lemma 1).

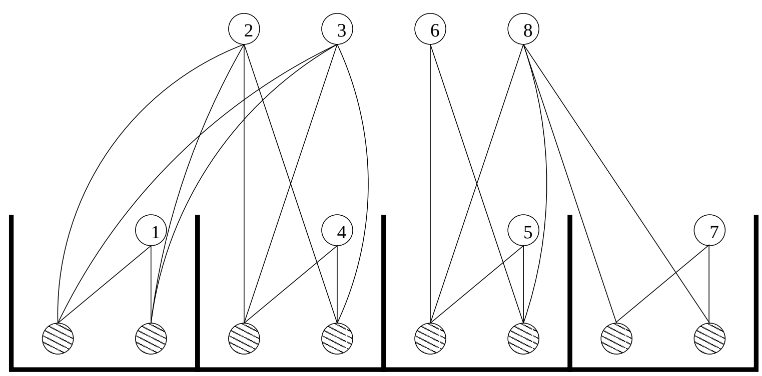


Figure 3: The vertices $\{1, 4, 5, 7\}$ are the randomly chosen roots. In the figure we see the right assignment of these vertices to the boxes. For all the neighbors of the special vertices we know the area of at most 3 boxes, where they belong to.

Now we construct an auxiliary-graph G_A in which each vertex of the input-graph is connected to the possible places in the boxes. Clearly a perfect matching in this graph gives us a layout (see figure 3). We have to describe the construction in more detail. The easiest method would be to build for each non-root vertex v the intersection B_v of the 3 surrounding boxes of all its roots. If there is some empty intersection, then the assignment of the roots to the boxes was wrong and we have to choose another one. Now we connect v to all the places in the boxes in B_v. If v is a root B_v is just the box where v was assigned to. It is easy to see that there is some assignment of the roots to the boxes, so that the perfect matching will give us a layout f with $B(f,G) \leq 6B(G)$.

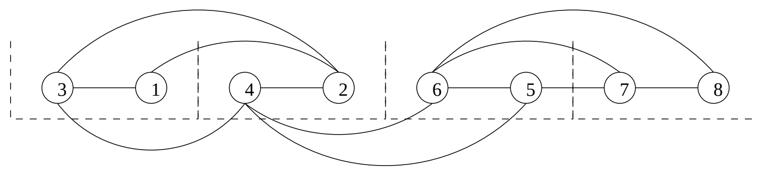
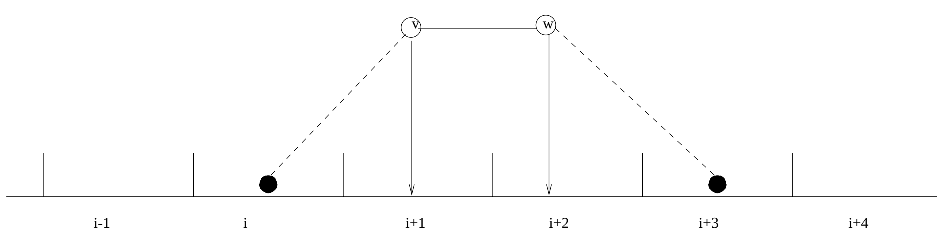


Figure 4: After running the perfect matching algorithm, we get a layout with maximum distance at most $3 \leq 2B(G)$.

To get a better approximation, we have to be more careful with the construction of B_v:

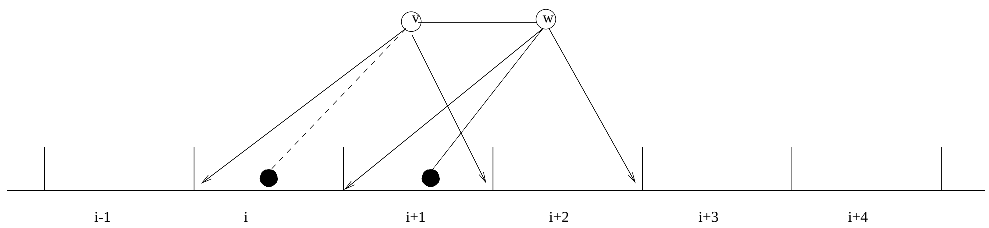
For each non-root vertex $v \in V$ and each root r_v of v we construct again a set of boxes and intersect them for all roots of v. Let r_v be assigned to box i. For each non-root neighbor w of v there are 4 possibilities (up to direction):

1. There is a root r_w of w in box $i+3$. Thus we know, that v has to be in box $i+1$ (and w in $i+2$). Otherwise v and w would be farer away than the bandwidth in any layout given by the perfect matching algorithm.

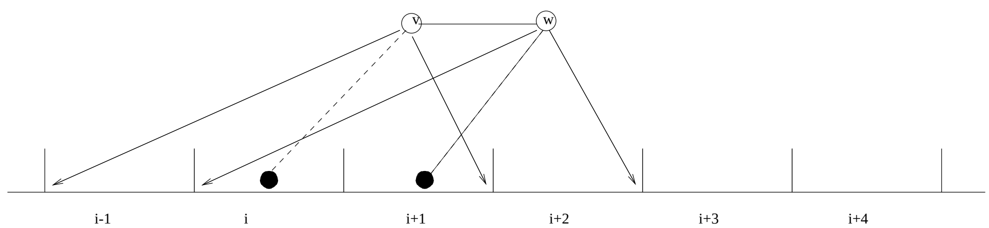
   The perfect matching can give us a layout f, in which v is on the left side of box $i+1$ and w on the right side of box $i+2$. Thus $B(f,G) \leq 2B(G)$.

2. w has its roots in the boxes $i+1$ and $i+2$. So v has to be in box i or $i+1$ (and w in $i+1$ or $i+2$).



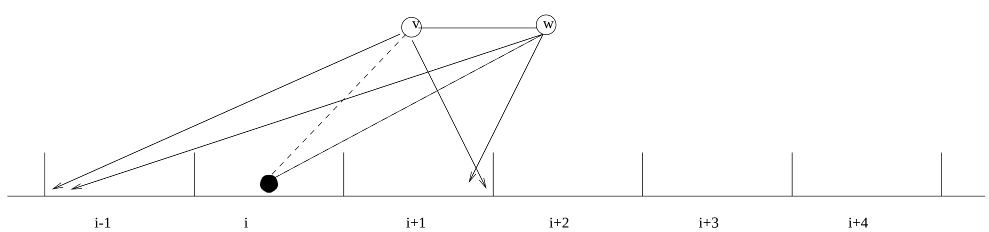
So the perfect matching will give us a layout f with $B(f,G) \leq 3B(G)$.

3. w has its roots only in $i+1$. v can be put into the boxes $i-1$ to $i+1$ (and w into i to $i+2$).



The worst-case arises in this case: The two non-roots v and w are adjacent. The dominating roots r_v and r_w lie in neighbored boxes i and $i+1$. The perfect matching algorithm assigns now u to the very left side of box $i-1$ and v to the very right side of box $i+2$.

4. All the roots of w are in box i. v can be put into the boxes $i-1$ to $i+1$ (and w into i to $i+1$).



Here we have for any layout $B(f,G) \leq 3B(G)$.

We can summarize the above discussion in the following algorithm.

```
Algorithm  DENSE_BANDWIDTH(G)
           { G is δ-dense }
           for  boxsize = δn  to  n/2  do
               begin
               { We have ⌈1/blocksize⌉ boxes, being parts of a layout}
               choose at random and independently a subset R ⊆ V of size O(log n);
               for  each possible assignment of the vertices of R to the boxes  do
                   begin
                   {build a bipartite auxiliary-graph G_A
                    of which one color-class consists of the places in the boxes
                    and the other class of the vertices of G}
                   for  each vertex v ∈ V  do
                       begin
                       Construct B_v
                       Connect v to all the places in the boxes of B_v
                       end
                   If there is a perfect matching in G_A, return one of them
                   { Note that a perfect matching M also defines a layout f_M}
                   end
               end
end  DENSE_BANDWIDTH
```

At least one of the polynomial number of assignments is correct and gives us a layout, which is not so far away from the optimum. Furthermore we can find a perfect matching in $O(|V||E|)$ time by the standard s-t-flow techniques [LP 86]. There are also some better methods [FM 91] [KR 97]. However this algorithm seems to be far away from being practical and the running time $PM(G)$ of the perfect matching algorithm will be dominated by the rest. We summarize our analysis in the following theorem:

Theorem 3 *There is a randomized algorithm which finds in $O(|V| \cdot |E| \cdot PM(G) \cdot \#(\textit{Assignments of the roots to the boxes}))$ bounded time for a δ-dense graph G a layout f, such that $B(f,G) \leq 4B(G)$.*

We can find the perfect matchings also in RNC [MVV 87] [KR 97]. It is easy, to construct the graph G_A in NC. So we have, by doing all the for-loops in parallel, the following theorem.

Theorem 4 *There is a RNC-algorithm which finds for a δ-dense graph G a layout f, so that $B(f,G) \leq 4B(G)$.*

3 Subdivisions for 3-Approximations

We can achieve a better approximation ratio by not using boxes of size $B(G)$, but a constant fraction of $B(G)$. We construct the bipartite auxiliary-graph G_A like in section 2. Our input graph G has minimum degree δn. Therefore $B(G) \geq \delta n$. In our improved algorithm we use boxes of size smaller than $\delta/2n$, which is a constant fraction of $B(G)$. Thus we have again a constant number of boxes and Lemma 2 remains true. We have much more boxes, so that the running time increases, but remains still polynomial for constant δ.

If the boxes have size $\delta/2n$, one can use an argument similar to the one used in Lemma 2 to show, that each vertex has in at least two different boxes of size $\delta/2n$ roots. Therefore we can isolate at most $2B(G)$ locations in the layout, where this vertex belongs to (see figure 5).

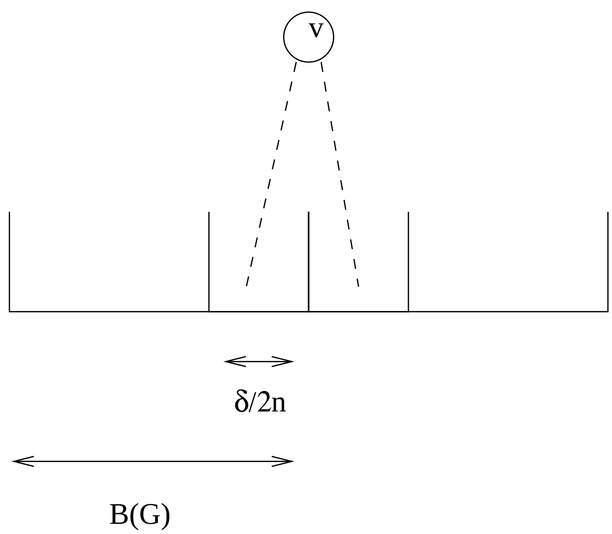


Figure 5: How to locate the $2B(G)$ locations in the layout.

If we have the area for a vertex v, we know that its neighbors must not be farer away then $B(G)$. That means that they are at most in the next $B(G)$ locations surrounding the area of v (see figure 6). We call the union of these areas the green area of v.

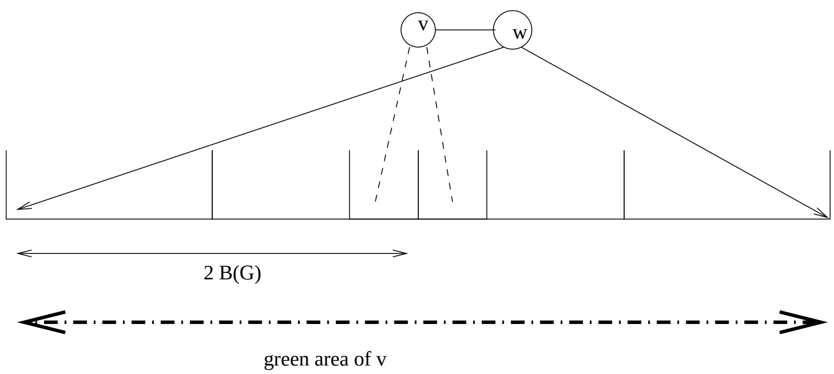


Figure 6: The green area of v.

If all neighbors of v located in this green area, then they are at most $3B(G)$ away from v. In order to compute B_v we build the intersection of v's area with the green areas of all its neighbors. This gives us

Theorem 5 *There is a randomized algorithm which finds in $n^{O(1/\delta)}$ time with high probability for a δ-dense graph G a layout f, so that $B(f,G) \leq 3B(G)$.*

As before we can parallize this algorithm.

Theorem 6 *There is a RNC-algorithm which finds for a δ-dense graph G a layout f, so that $B(f,G) \leq 3B(G)$.*

4 The Bandwidth Problem in Directed Graphs

In this section we will present a 2-approximation algorithm for the directed δ-dense bandwidth problem. We call a directed graph G δ-dense, if each vertex has in-degree at least δn.

The bandwidth problem in this case is similar to the undirected case, except for the restriction that all incoming edges of a vertex v has to lie on the left hand side of v in any valid layout. Thus we can only find a valid layout for acyclic graphs.

Like in Section 3 we use boxes of size $\delta/2n$. Like before R will build a dominating set (each non-root vertex v has an incoming edge with one endpoint in R), such that v has roots in different boxes. If v has a root in box i and j ($i < j$) (of size $\delta/2n$) v can only be connected to the places of the boxes which are $B(G)$ on the right of j including j (see figure 7).

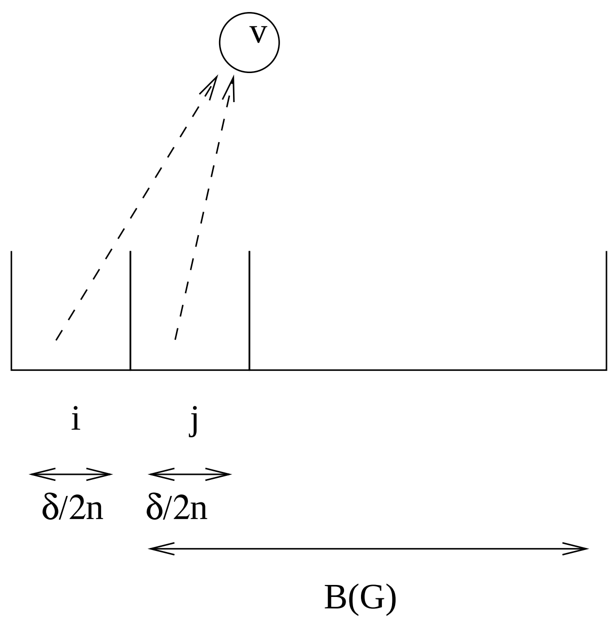


Figure 7: Possible placement for v.

For v we know now that all its neighbors are only allowed to be assigned to the $2B(G)$ places on the left of box j. This area is again called the green area. If we find a layout f in which all the neighbors of each vertex are assigned to their green areas, then we have

$$B(f,G) \leq 2B(G)$$

We build G_A like in section 3 and get since we have still a constant number of boxes the following results.

Theorem 7 *There is a randomized algorithm which finds in $n^{O(1/\delta)}$ time with high probability for a δ-dense directed acyclic graph G a layout f, so that $B(f,G) \leq 2B(G)$.*

As before we can parallize this algorithm.

Theorem 8 *There is a RNC-algorithm which finds for a δ-dense directed acyclic graph G a layout f, so that $B(f,G) \leq 2B(G)$.*

5 The Superdense Case

Further densification leads to polynomial approximation scheme for the bandwidth minimization problem. We call a simple graph G superdense, if the minimum degree of G is at least $n - o(n^\delta)$. The notion of superdenseness has been introduced in [KZ 97].

If G is superdense, $B(G)$ is at least $n - o(n^\delta)$ and therefore any layout f will suffice to be a good approximation:

$$\begin{aligned} B(f,G) &\leq n \\ &\leq (1+\epsilon)n - o(n^\delta) \quad \text{for any } \epsilon \in O(1) \\ &\leq (1+\epsilon)(n - o(n^\delta)) \\ &\leq (1+\epsilon)B(G) \end{aligned}$$

6 Further Research and Open Problems

There remains still an important open problem of designing polynomial time approximation algorithms with approximation ratio less then three for the bandwidth problem on dense graphs. More strongly, can we hope that a PTAS exists for this problem or is the problem MAX-SNP-hard (see [PY 88])? At the moment we do not know whether the general bandwidth problem is MAX-SNP-hard nor whether the bandwidth for dense graphs is in fact NP-hard.

We were not able to prove any constant ratio approximation of the bandwidth for dense *in average* graphs (see [AKK 95]) having $\Theta(n^2)$ edges. For this case however we were able to prove its NP-hardness.

Acknowledgment

We thank Sanjeev Arora, Haim Kaplan and Uri Zwick for helpful discussions.

References

[AKK 95] Arora, S., Karger, D., Karpinski, M., *Polynomial Time Approximation Schemes for Dense Instances of NP-Hard Problems*, Proc. 36^{th} ACM STOC (1995), pp. 284–293.

[CCDG 82] Chinn, P., Chvatalova, J., Dewdney, A., Gibbs, N., *The Bandwidth Problem for Graphs and Matrices - A Survey*, Journal of Graph Theory (1982), pp. 223–254.

[FM 91] Feder, T., Motwani, R., *Clique Partitions, Graph Compression and Speeding-up Algorithms*, Proc. 23^{rd} ACM STOC (1991), pp. 122–133.

[GGJK 78] Garey, M., Graham, R., Johnson, D., Knuth, D., *Complexity Results For Bandwidth Minimization*, SIAM J. Appl. Math. **34** (1978), pp. 477–495.

[HMM 91] Haralamides, J., Makedon, F., Monien, B., *Bandwidth minimization: an approximation algorithm for catapillars*, Math. Systems Theory **24** (1991), pp. 169–177.

[KR 97] Karpinski, M., Rytter, W., *Fast Parallel Algorithms for Graph Matching Problems*, Oxford University Press, 1997.

[KZ 97] Karpinski, M., Zelikovsky, A., *Approximating Dense Cases of Covering Problems*, Technical Report TR-97-004, ECCC, 1997.

[Kr 87] Kratsch, D., *Finding the Minimum Bandwidth of an Interval Graph*, Information and Computing **74** (1987), pp. 140–187.

[LP 86] Lovasz, L., Plummer, M., *Matching Theory*, North-Holland, Amsterdam, 1986.

[MVV 87] Mulmuley, K., Vazirani, U. V., Vazirani, V. V., *Matching is as Easy as Matrix Inversion*, Proc. 19^{th} ACM STOC (1987), pp. 345–354.

[Pa 76] Papadimitriou, C., *The NP-Completness of the Bandwidth Minimization Problem*, Computing **16** (1976), pp. 263–270.

[PY 88] Papadimitriou, C., Yannakakis, M., *Optimization, Approximation, and Complexity Classes*, Proc. 20^{th} ACM STOC (1988), pp. 229–234.

[PS 97] Prömel, H. J., Steger, A., *RNC-Approximation Algorithm for the Steiner Problem*, Proc. STACS'97 (1997).

[Sa 80] Saxe, J., *Dynamic programming algorithms for recognizing small-bandwidth graphs*, SIAM Journal on Algebraic Methods **1** (1980), pp. 363–369.

[Sm 95] Smithline, L., *Bandwidth of the complete k-ary tree*, Discrete Mathematics **142** (1995), pp. 203–212.

[Tr 97] Trevisan, L., *Positive Linear Programming, Parallel Approximation, and PCP's*, Proc. 4^{th} European Symposium on Algorithms (1997).

[TX 97] Trevisan, L., Xhafa, F., *The Parallel Complexity of Positive Linear Programming*, submitted (1997).