DEPARTMENT OF COMPUTER SCIENCE
FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS
COMENIUS UNIVERSITY, BRATISLAVA

ALGORITHMS AND HEURISTICS FOR ANTIBANDWIDTH PROBLEM OF BIPARTITE GRAPHS

(master's thesis)

2007

Peter Petrovský

DEPARTMENT OF COMPUTER SCIENCE
FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS
COMENIUS UNIVERSITY, BRATISLAVA

ALGORITHMS AND HEURISTICS FOR ANTIBANDWIDTH PROBLEM OF BIPARTITE GRAPHS

(master's thesis)

PETER PETROVSKÝ

RNDr. Imrich Vrt'o, DrSc.
Advisor

May 2007, Bratislava

I hereby declare that this thesis is my own work, only with the help of the referenced literature and under the supervision of my thesis advisor.

. . . . . .

Peter Petrovský

Acknowledgements

I would like to thank my thesis advisor RNDr. Imrich Vrt'o, DrSc. for careful guidance, useful advices and helpful discussions during the work on this thesis and to my parents, family and friends for their love and support.

Abstract

The *antibandwidth* problem for graph G consists of labelling n vertices v_i of the graph with distinct integers $f(v_i)$ in range $[1, n]$ in such manner that value

$$min\{|f(v_i) - f(v_j)| : (v_i v_j) \in E(G)\}$$

is maximized over all labellings. The problem was originally introduced like dual variation of well-known *bandwidth* problem, but it can be reinterpreted in many ways – as special multiprocessor scheduling problem , special linear layout problem or variant of obnoxious facility location problem. The *antibandwidth* problem is NP-complete, there are very few exact results for nontrivial graph classes and some classes of graphs where time for finding the parameter is polynomially bounded. No general heuristics are known so far. In this paper, we give necessary overview of all fundamental information about the problem and then introduce idea of general heuristic for obtaining constructive lower bounds of *antibandwidth* parameter of arbitrary bipartite graphs. We describe some variations in the technique and their influence on the results and efficiency of the algorithm. We give statistical comparison to few known results and analyze reasonability for using proposed heuristic.

Keywords: graph theory, antibandwidth, dual bandwidth

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The origin of interest about antibandwidth problem can be found in [11]. Motivation in this work was to explore some variants of the bandwidth minimization problem, which is usually presented in following form: For a graph G the problem is to label the n vertices v_i of G with distinct integers $f(v_i)$ from $\{1, 2, \ldots, n\}$ so that the value

$$max\{|f(v_i - f(v_j))| : (v_i v_j) \in E(G)\}$$

is minimized over all labellings. In contrast to antibandwidth problem which is its dual form, research on bandwidth problem was induced from practical needs in engineering. In 1950s structural engineers first analyzed steel frameworks by computer manipulation of their structural matrices. In order that operations like inversion and finding determinants take as little time as possible, the attempt was made to discover an equivalent matrix in which all the nonzero entries lay within a narrow band about the main diagonal — hence the term *bandwidth* [3]. In exact form problem was as follows. For a real symmetric matrix M to find a symmetric permutation M' of M so that the maximum value of

$$|i - j|$$

taken over all nonzero entries m'_{ij} is minimal. The equivalence of these two problems is made clear by replacing the nonzero entries of M by 1's

and interpreting the result as the adjacency matrix of a graph. Then the labelling of the graph is equivalent to symmetrical permutation of its adjacency matrix. It can be interesting that bandwidth problem for graphs, meanwhile, originated independently, but also from practical needs of theory of coding and searching for codes which minimize the maximum error.

It can be seen that despite this two problems are closely related in mathematical sense, naturally more attention was paid to bandwidth problem. It is much more explored and there are numerous papers on various aspects of problem. In context of the fact, that both of these problems are NP-complete, especially important point is, that there exists heuristic called Cuthill-McKee [4] for obtaining reasonable estimates of bandwidth parameter for arbitrary graph. The technique of implementation improvements and analysis of algorithm complexity and quality of results can be found in various works e.g. [6]. Algorithm for performing this estimate is widely used in applied field and is often included in standard mathematical matrix transformation libraries. On the other hand, no such general heuristic is known for antibadwidth problem, even for such important subclass like bipartite graphs.

1.1 Motivation

Although the original problem arose like a variation of bandwidth problem, today there are more direct motivations for solving it at least approximately, finding another classes of graphs where there can be produced exact results or classes of graphs, where polynomial algorithms or reasonable heuristics can be used. According to [16], basic motivations are these: First of them to consider is radio frequency assignment problem. The problem is to assign n different frequencies to n different transmitters in such manner that physically neighbouring transmitters have as different frequencies as possible [8]. The transmitters are represented by the vertices of a graph G and their frequency neighbourhood is represented

by the adjacency in graph G. This is a special interpretation of more general obnoxious facility location problems [2]. Given an "enemy graph", vertices of which are some entities and edges represent relation of their mutual intolerance, task is to arrange them on the line so that the minimal distance between any couple of enemies is maximized. This is a special case of more complex so-called antidilatation problem, where the "host graph" is a path. In general we can try to embed guest enemy graph into arbitrary graph structure with intention to maximize minimal distance between vertices of embedded guest graph. There are many interesting applications of this general problem in various fields, e.g. when the host graph is a hypercube Q_n and the guest graph is a complete graph K_p [14]. Then the solution for antidilatation problem for $p \leq 2^n$ is equal to Hamming distance of a binary code with p words of length n, which is the basic parameter in theory of coding to determine error correcting property of given code.

Scheduling problems are another area, where more information about antibandwidth problem can be applied. E.g. when given a graph of tournament where players are represented by vertices and planned matches by adjacency in this graph, we need to schedule these games in such way, that time breaks between any two games involving same players are longest as possible thus giving players maximal rest. Although this case requires transformation of the tournament graph to kind of corresponding edge graph before the solution becomes eqivalent to finding antibandwidth parameter, idea of usage remains the same. Similar problem can be considered in area of multiprocessor or multitask scheduling [11], where we require some related problems sharing particular resources to be computed in biggest possible time or space distance as possible.

Last field of motivation for finding new results about antibandwidth parameter to mention, is from area of interconnection networks [16]. Since there has been big advance in VLSI technology in recent years, interconnection networks are becoming very complex, focusing more on integra-

tion of big number of computational units rather than increasing their computational power. The overhead of communication between them is becoming very complex and one important problem is to find proper topological structure for the network satisfying various requirements. When thinking about one dimensional (linear) layout, we are again in touch with antibandwidth problem [16].

1.2 Goals and results

The goal of this work is to investigate ideas that could lead to finding heuristics or algorithms for computing reasonable lower bounds of antibandwidth parameter of arbitrary bipartite graphs. Complexity of the problem for bipartite graphs is not known and no algorithms exist either. After summarizing known issues about the problem focusing on algorithmic and computational part, we propose idea of one such heuristic. We analyze its parameters and how their selection affects the results and efficiency of the algorithm. Then we give statistical comparison of results obtained by our implementation of the heuristic. Even though testing the quality of heuristic is problem because of non-availability of many reference results, for existing ones we compare obtained values and discuss the convenience of heuristic for every particular tested class. We give results also for some examples of random bipartite graphs where no reference results are known and so our values are first available general approximations for this cases. Finally we offer algorithmical and complexity analysis of the heuristic.

Chapter 2

Preliminaries, notation and definitions

2.1 Terminology

In this place, there is a need to make a basic terminology of problem clearer, because there are some inconsistencies in naming the same problem in various papers. The original paper [11], which in fact introduced the problem, was using the name *separation number* of graph. But during recent years this name was adopted for another kind of graph linear layout problem, where it seemed more natural. In another paper [12] Lin and Yuan are introducing the same problem under term *dual bandwidth*. Finally, in work [15] the term *antibandwidth* is proposed and we will use this one, because it seems most appropriate and least confusing.

2.2 Notation

We use the following notation and conventions. Only finite simple nonoriented graphs G with vertex set $V(G)$ and edge set $E(G)$ will be considered. As above let $[1, n]$ denote the set of integers from 1 to n inclusive. Let $G \subseteq H$ denote that G is isomorphic to a subgraph of H. We let $\lceil x \rceil$, and $\lfloor x \rfloor$

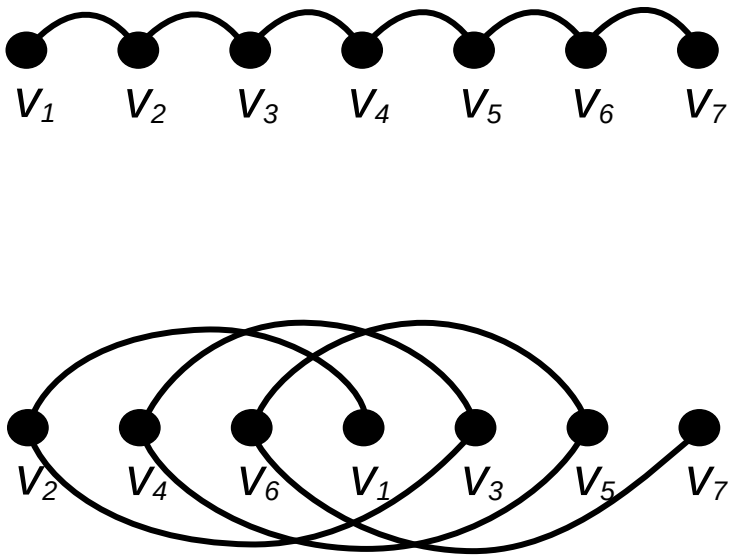

Figure 2.1: Optimal antibandwidth layout for P_7. Formal labelling of vertices is $f(v_1) = 4$, $f(v_2) = 1$, $f(v_3) = 5$, $f(v_4) = 2$, $f(v_5) = 6$, $f(v_6) = 3$, $f(v_7) = 7$

be the usual ceiling and floor functions, i.e., the smallest (resp. greatest) integer greater then (resp. less than) or equal to x. We further use common conventions like $\delta(G)$ for smallest degree of a graph, $\Delta(G)$ for the largest degree and $D(G)$ for graph diameter. $\chi(G)$ is used for chromatic number of graph, $\alpha(G)$ for independence number. $\overline{G}$ stands for graph which is the complement of G.

2.3 Definitions

Definition 2.3.1 (Antibandwidth parameter) *For a nonempty graph $G = (V, E)$ let f be a one-to-one labelling*

$$f : V \to \{1, 2, \ldots |V|\}$$

We define the antibandwidth of G according to f as

$$ab(G, f) = \min_{uv \in E(G)} |f(u) - f(v)|$$

The antibandwidth of G is defined as

$$ab(G) = \max_f ab(G, f)$$

It is useful to imagine the antibandwidth problem also as a linear layout problem. The labelling maps vertices of the graph on a line of integer points in range $[1, n]$ in such manner that minimal distance of adjacent vertices is maximized and so "spreading the graph from inside" as much as possible. Simple example of the optimal antibandwidth layout of P_7 is on Figure 2.1.

Definition 2.3.2 (k^{th} power of a graph) *The k^{th} power of a graph $G = (V, E)$ is the graph G^k with the same vertex set and $xy \in E(G^k)$ if and only if the distance between x and y in G is at most k.*

Later in the chapter we are considering k^{th} powers of Hamiltonian paths contained in other graphs. We say that G contains the k^{th} power of a Hamiltonian path if $|V| = n$ and if G has a subgraph isomorphic to P_n^k, where P_n is the path on n vertices. In terms of labelling, this is equivalent to following: G contains the k^{th} power of a Hamiltonian path if the vertices can be labelled by unique integers from $[1, n]$ in such way that $|i - j| \leq k$ implies $ij \in E(G)$.

Chapter 3

Previous results

In this chapter we summarize known results on the problem. As we mentioned in the introduction, there are few results that can be considered significant. Anyway, we give an overview of them focusing on the ones, that have importance to subject of this thesis. Since the problem is (like we show later in this section) NP-complete, any reasonable estimation can be an advance. From our point of view, so far known results can be roughly divided into three main sections.

First direction of research was to connect antibandwidth problem with other well-known and well-studied graph invariants and characteristics like $\chi(G)$, $\Delta(G)$ or $\delta(G)$ which brought some basic facts about upper and lower bounds. Special case is the connection with its dual and closely related bandwidth problem. This one is much better surveyed, so any practical result from this connection would be requisite. Nothing strong has been proven so far.

Another set of theorems is focusing on particular (and sometimes very narrow) classes of graphs, where exact results (usually lower bounds) are proven, based on finding particular labelling. These include results for paths, complete k-ary trees, hypercubes, meshes ...

The last group of results are computational issues. These set of theorems proves NP-completness of problem, even of some subproblems,

complementarity with the problem of finding powers of Hamiltonian path. Further it surveys graph classes, where polynomial algorithms are applicable.

3.1 Basic bounds

We start with basic upper and lower bounds and connection to other invariants. For closer look we recommend [13].

Theorem 3.1.1 *For connected graph $G = (V, E)$ of order n*

$$ab(G) \leq \left\lfloor \frac{n}{2} \right\rfloor$$

We could see that this absolute possible upper bound can be reached e.g. in case of paths P_n (Figure 2.1). It comes from a simple observation that it is not possible for all edges to have length greater than $\frac{n}{2}$ in any linear layout. The optimal labelling for paths $P_n = (v_1 v_2 \dots v_n)$ can be formalized as

$$\begin{aligned} f(v_i) &= \frac{i}{2} \quad for\ i\ even \\ f(v_i) &= \left\lceil \frac{n+i}{2} \right\rceil \quad for\ i\ odd \end{aligned} \tag{3.1}$$

Obviously $ab(G)$ can have a value of 1 (e.g. for complete graphs K_n) so the basic bounds of antibandwidth values for connected graphs of order n are $[1, \lfloor \frac{n}{2} \rfloor]$.

The known relations between antibandwidth parameter and other graph invariants are presented here. They mainly present upper bounds which can bring some theoretical information but of course, since we are searching for the best possible estimate, lower bounds and especially constructive ones are really what we are looking for. These are usually obtained by (ad hoc) constructions which are usually dependent on property of some particular class. For closer look and proofs we recommend [13] resp. [12].

Theorem 3.1.2 *For connected graph $G = (V, E)$ with n vertices and q edges*

(i) $ab(G) \leq n + \frac{1}{2}(1 - \sqrt{1 + 8q})$

(ii) $ab(G) < \frac{n}{\chi(G)-1}$

(iii) $ab(G) \leq \frac{1}{2}(n - \delta + 1)$

(iv) $ab(G) \leq n - \Delta$

(v) $ab(G) \leq \alpha(G)$

(vi) for G bipartite, $ab(G) \geq \left\lceil \frac{D(G)}{2} \right\rceil$

So far, we have not spoken much about the connectivity of the examined graph. According to our definition 2.3.1, things are all right when the graph is not connected, only if there is no edge at all parameter is undefined and we can fix it to ∞ by definition. Non-connectivity gives us chance to even overcome the upper bound of $\frac{n}{2}$ for graphs with $\delta(G) > 0$. Interesting question can be, how does the overall antibandwidth depend on antibandwidth of its components and if it can be constructed from them. This problem seems to remain hard but this approach, specialized on the copies of the same graph (considered as union) was studied in work [7]. Some of the results can be inspiring for our later heuristic, especially regarding the non-connected graphs and combining their vertices from components to produce good antibandwidth estimate, like in the following ones. Constructive proofs can be found in mentioned work [7].

Theorem 3.1.3 *For graph G and $k \in \mathbb{N}$, $ab(k \cdot G) \geq k \cdot ab(G)$.*

Theorem 3.1.4 *Let G be the union of paths P_i with $i = q_j > 0$ (j=1,...,m) and $\sum_{i=1}^{m} q_i = n$. Then $ab(G) = \lfloor n/2 \rfloor$.*

3.2 Lower bounds and exact results

There are only few results giving exact numbers or at least asymptotical lower bounds. Most of them are quite narrow special graph classes. Reason is that unless G has some nice regular structure, it is really hard to give some general description how to label the vertices in sense of good antibandwidth and prove value of lower bound. One of the biggest results in this field is work [13] which offers outline how to proceed in construction of the labelling for simple subclasses of bipartite graphs and determine antibandwidth of this labelling afterwards. Further it uses this technique to prove reasonable lower bounds of antibandwidth for forests, special trees, meshes and hypercubes. Works [15] and [16] extend the results and give also upper bounds for these cases. We will use these results later on for comparison with our estimates, so here is the list of the main issues. More information and proofs can be found in above referenced articles.

Theorem 3.2.1 *Let F be general forest, $MIN(F)$ the minor bipartition of F. Then $ab(F) \geq MIN(F)$.*

Proof of this theorem is constructive and gives algorithm that arranges vertices of tree with resulting antibandwidth of smaller bipartition.

Theorem 3.2.2 *For F a forest, $ab(F) = \lfloor \frac{n}{2} \rfloor$ if and only if F is balanced or contains a vertex of degree 1 or 2 whose removal yiels a balanced forest.*

The key theorem, which stands back behind most of theoretical results for bipartite graphs follows.

For bipartite graph B with bipartition X, Y with $|X| = m$, $|Y| = n - m$, with $Y = \{y_1, y_2, \ldots y_{n-m}\}$ (fixed order), consider all labellings $f : V(B) \rightarrow [1, n]$ for which $f(y_i) = m + i$ for all $i \in [1, n - m]$. Let $ab(B, \{y_i\})$ be the maximum value of $ab(B, f)$ among all such labellings f. For each $x \in X$, let $j(x) = min\{i : xy_i \in E(B)\}$. For each $t \in [1, n-m]$, let $Nb(t) = \{x \in X : j(x) = t\}$, let $New(t) = |Nb(t)|$ and let $B(i) = \sum_{t=1}^{i} New(t)$, the number of vertices adjacent to any or all elements of $\{y_1, y_2, \ldots y_i\}$.

Theorem 3.2.3 $ab(B, \{y_i\}) = m - max_i\{B(i) - i\}$

Theorem gives us method how to determine antibandwidth value of particular labelling for bipartite graphs. If we choose this labelling "smart", summation can lead to a optimal lower bound in closed or asymptotic form. Here are the known outcomes, some improved not to be only lower bounds, but exact values [15] [13] [16].

Theorem 3.2.4 *Let P_n denotes path of length n, C_n cycle of length n, $T_{k,n}$ complete k-ary tree with total number n of vertices, $P_m \times P_n$ denotes mesh graph of dimension $m \times n$ with $m \geq n$ and Q_n the hypercube of order n. Then*

(i)
$$ab(P_n) = \left\lfloor \frac{n}{2} \right\rfloor$$

(ii)
$$ab(C_n) = \left\lceil \frac{n}{2} \right\rceil - 1$$

(iii) For even $k \geq 4$

$$ab(T_{k,n}) = \frac{n + 1 - k}{2}$$

For odd $k \geq 3$ and $h \geq 3$, h is the height of the tree

$$ab(T_{k,n}) \geq \frac{n}{2} - O(k^2 h)$$

(iv)
$$ab(P_m \times P_n) = \left\lceil \frac{n(m-1)}{2} \right\rceil$$

(v)
$$ab(Q_n) = 2^{n-1} - \frac{2^n}{\sqrt{2\pi n}}(1 + o(1))$$

(vi) $n \geq 3$, e *is whichever of* $\lfloor \frac{n-1}{2} \rfloor$ *or* $\lfloor \frac{n+1}{2} \rfloor is\ even$

Let

$$F(e,n) = \binom{n}{e+1} + \binom{2e}{e} - \binom{n}{e} - \binom{2e}{e+1} - (n-e-1) + \sum_{j=1}^{e-1} \left[\binom{2e-2j}{e-j} - \binom{2e-2j}{e-j+1} \right]$$

Then

$$ab(Q_n) \geq 2^{n-1} - \left[\binom{n-1}{e-1} + (n-e-1) + F(e-1, n-1) \right]$$

if $n \equiv 3\ (mod\ 4)$

and

$$ab(Q_n) \geq 2^{n-1} - \left[\binom{n-1}{e-1} + (n-e-1) + F(e,n) \right]$$

otherwise

Moreover this bound is asymptotically optimal

3.3 Computational results

In this section, we are going to look closer on the computational issues of antibandwidth problem, its complexity and known algorithmic results. Antibandwidth problem is proven to be hard and therefore no polynomial time algorithm can be used for solving it efficiently. More on this and the proofs of theorems in this section can be found in [11] and [12]. Following theorem shows, that even a subproblem is very hard to be solved.

Theorem 3.3.1 $ab(G) \geq 2$ *if and only if* $\overline{G}$ *has a Hamiltonian path.*

It is useful thing to realize, how directly is our problem complementary to well-known Hamiltonian path problem. Let $ab(G) \geq 2$ and let f is optimal antibandwidth labelling for G. Let $v_1, v_2, \ldots, v_n$ is ordering of vertices of G such that $f(v_i) = i$. According to $ab(G) \geq 2$, there cannot be any edge

between v_i and v_{i+1} for $i = 1, 2, \ldots, n-1$. Thus sequence $v_1, v_2, \ldots, v_n$ is defining Hamiltonian path in $\overline{G}$.

Conversly, let $v_1, v_2, \ldots, v_n$ is the Hamiltonian path in $\overline{G}$. If we define the labelling of G simply like $f(v_i) = i$ then, because there are no edges between neighbours in linear layout $ab(G) \geq ab(G, f) \geq 2$.

The simple corollary of this result is

Theorem 3.3.2 *Decision problem "is $ab(G) \geq 2$" for given graph G is NP-complete.*

Proof comes out from the complementarity of problem $ab(G) \geq 2$ to problem of finding a Hamiltonian path in G (showed in the last Theorem 3.3.1) and the NP-completness of Hamiltonian path problem.

This result shows the root of the problem – even this simple decision is in general practically incomputable and that is the real motivation why we are searching at least for some reasonable constructive lower bounds of the parameter.

Analogically to above, it is easy to see that for graph G of order n problems if $ab(G) \geq k$ and $\overline{G}$ contains k^{th} power of Hamiltonian path P_n (Definition 2.3.2) are equivalent. Moreover there are some graph classes, where the maximum k such that graph of this class contains k^{th} power of Hamiltonian path (Hamiltonian path power problem) can be detected by particular algorithms in polynomial time. That gives us polynomial detection of antibandwidth parameter for complementary classes.

Theorem 3.3.3 *There are efficient algorithms for finding antibandwidth parameter in complements of*

(i) interval graphs

(ii) treshold graphs

(iii) arborescent comparability graphs

All necessary definitions of these graph classes, theorems, proofs and algorithms can be found in [10] resp. [5].

No complexity issues are known so far for the case of bipartite graphs. There is no similar proof that problem is hard like the above ones, nor any known polynomial algorithm solving the problem.

Chapter 4

Heuristic for bipartite graphs

As mentioned in previous chapter, complexity of finding antibandwidth for arbitrary bipartite graphs is not known. Despite this fact, property of being bipartite and so offering independent sets of vertices with no edges within is adding some convenience into heuristical search of parameter approximation. As we can see from (ii) in Theorem 3.1.2, bipartite graphs (synonym for having $\chi(G) = 2$) are interesting also for being only class that can achieve all possible values of antibandwidth parameter from 1 (for complete bipartite graphs) to $\lfloor \frac{n}{2} \rfloor$ (e.g. for path graph). We could see in previous sections that there are some theoretical results that make use of the fact that graph is bipartite.Work [13] introduces reasonable solution for forests based on algorithmical approach and Theorem 3.2.3 gives advice how to proceed in showing lower bounds of antibandwidth for particular labelling of bipartite graph. Usually very nice and regular labelling for kind of regular bipartite graph have to be used in order to be able to calculate the summation given by theorem. Good examples of this method are results from Theorem 3.2.4 for hypercubes or meshes.

Our aim here is different - we want to construct general heuristic for constructing lower bounds that can just benefit from knowledge of graph being bipartite, but nothing more. Obviously the results therefore cannot tend to be optimal, but since there is not any known general heuristic,

everything producing reasonable results can be analyzed.

4.1 Basic observations and ideas

First we can look on simple general observations of properties of our problem that could help us with the construction. We can be inspired from different heuristics for, at first glance, related problems. One of them can be *generalized maximum linear arrangement problem* [9]. That is to compute for a given vector $x \in \mathbb{R}^n$ and $n \times n$ non-negative symmetric matrix $W = (w_{i,j})$ permutation π of $\{1, 2, \ldots, n\}$ that maximizes $\sum_{i,j} w_{\pi_i,\pi_j}|x_j - x_i|$. If we set vector x to $(1, 2, \ldots, n)$, it is special NP-hard case known as *maximum linear arrangement*. If we take matrix W for binary incidence matrix of graph with vector x set to $(1, 2, \ldots, n)$ we have something what we can call "average antibandwidth problem" - we try to maximize sum of all edge distances in linear layout given by permutation π, while in common antibandwidth problem we just want to maximize shortest one. For the average antibandwidth problem, we can use very simple heuristic method. Since the average value of $|x_j - x_i|$ in this case is $\frac{n}{3}$, expected weight of a random permutation is $\frac{n}{3}\sum_{i,j} w_{\pi_i,\pi_j}$, which is at least $\frac{1}{3}optimal$ in expected case [9].

We can look what random permutation would produce in common antibandwidth problem. For purpose of simplicity, let us look on the case of random graph on n vertices, with independent probability p for each possible edge to exist.

Lemma 4.1.1 *For $G_{n,p}$ random graph on n vertices with edge probability $p \in (0, 1)$, random labelling f of vertices almost always produces $ab(G, f) = 1$ for any p.*

Proof. Let us imagine linear layout mapping of G. Let f is random permutation, that maps some vertex v to the first point of linear layout. Probability that neighbour in the linear layout is not connected to v in G is $1-p$

(like for any other vertex). Consequently, probability that there is no edge between any neighbour vertices in linear layout is $(1-p)^{n-1}$. That means that possibility of $ab(G, f) \geq 2$ with rising n approach to 0 for any p. □

Randomness in this way does not seem to be very helpful approach in our case.

Another problem property that can usually mean some benefit for finding reasonable heuristic is "continuity" or "stability" or in the other words that small change in the domain of problem cause only small change in the result. This can lead to iterative improvement of temporary results based on some decision algorithm. If we take the mutual change of the labels of two vertices for minimal change in the domain, problem is not continuous because this elementar change can obviously lead from best possible result to the worst (consider the change of positions of vertices v_4 and v_1 on Figure 2.1). If we take the mutual change of two neighbouring labels (that differ by 1) of the vertices for minimal change in the domain, it can lead to change of antibandwidth by 1 to each side or no change at all. On the first look, antibandwidth problem must be considered as global problem, because any locally good situation can still mean worst result overall. This property indicates that just local changes, which can be relatively under control, may not lead to good overall result and big changes of global influence are also needed, but control over them is quite lost with high number of edges, because they can always lead to the worst case. The heuristic working with the motivation of iterative improvements should consider this fact and so maybe some concept of generic probabilistic meta-algorithm for the global optimization allowing big changes at least in the beginning, like *simulated annealing*, could bring some success. But so far no issue is known regarding this direction.

4.2 Description of heuristic

Basic idea of proposed heuristic is that bipartite graph can be easily decomposed to number of independent sets of vertices - not having any edges within. Then we can arrange them lineary in such order, that any existing edges between these sets are tried to be maximized by letting them "go over" most possible other sets, where no edges are attached inside. In the following we suppose G to be connected. If it is not, we are making the same procedure for all of its components and then construct final one afterwards as will be described later below.

Algorithm will be like follows. Let G is arbitrary bipartite graph. We choose one vertex or independent set of vertices in it (randomly or with some strategy resp.), which we will call seed. Then we construct decomposition $D_{seed}(G)$ of G to set of vertex layers, beginning with the seed, N stands for neighbours.

$$D_{seed}(G) = \{L_1, L_2, \ldots, L_m\}$$

$$V(G) = \bigcup_i L_i$$

$$L_i \bigcap_{i \neq j} L_j = \emptyset$$

$$L_1 = \{seed\}$$

$$L_{k+1} = N(L_k) \setminus \bigcup_{i<k} L_i$$

or in non-inductive way

$$L_k = \{u \in V(G) : s \in seed, d(s,u) = k-1\}$$

Important property of this decomposition is that it guarantees the edges to exist just between neighbouring layers L_i and L_{i+1} for $i \in [1, m-1]$ and nowhere else due to the construction and the fact that G is bipartite. We can use this "linearity" of decomposition for maximizing existing edges in the same manner as with the path graph P_n. Example of such a labelling is

on Figure 2.1. It is optimal antibandwidth labelling for path giving highest possible value of $\lfloor \frac{n}{2} \rfloor$ and the assigning labels to vertices is given by formula (3.1).

We will use the same technique for ordering the set of n layers of our decomposition D. Let us consider labelling of layers $g : D \rightarrow [1, n]$

$$g(L_i) = \frac{i}{2} \quad for\ i\ even$$
$$g(L_i) = \left\lceil \frac{n+i}{2} \right\rceil \quad for\ i\ odd \tag{4.1}$$

Vertices inside layers will be temporarily labelled in natural order or randomly, we will discuss strategies on this point and their effect on result later in section 4.3.2. Seed set for this moment is chosen as random vertex, strategies of better selection will follow in section 4.3.1. Picture with schematic demonstration of the procedure is on the Figure 4.1.

Given a graph decomposition $D(G)$, the minimal width $mw(D)$, maximal width $w(D)$ and depth $d(D)$ stand for $min_i|L_i|$, $max_i|L_i|$ and the number of layers respectively. By numbering G arbitrarily, layer by layer, it is not hard to see that for such a numbering f

$$ab(G) \geq ab(G, f) \geq mw(D)(\left\lfloor \frac{d(D)}{2} \right\rfloor - 1) + 1 \tag{4.2}$$

and

$$ab(G, f) \leq w(D)(\left\lfloor \frac{d(D)}{2} \right\rfloor - 1) + 1 \tag{4.3}$$

From this basic relation, we can see that the crucial factor for obtaining best possible result is to maximize product of $mw(D)$ and $d(D)$ by various choices during heuristic.

We can express the lower bound of constructed antibandwidth estimate even in more complex and precise way. The value, which is the shortest possible edge length in the constructed layout, is determined by the minimal sum of cardinalities of the $\lfloor \frac{d(D)}{2} \rfloor - 1$ consecutive layers, which are together overrun by edges of the graph (Figure 4.2). Length of the

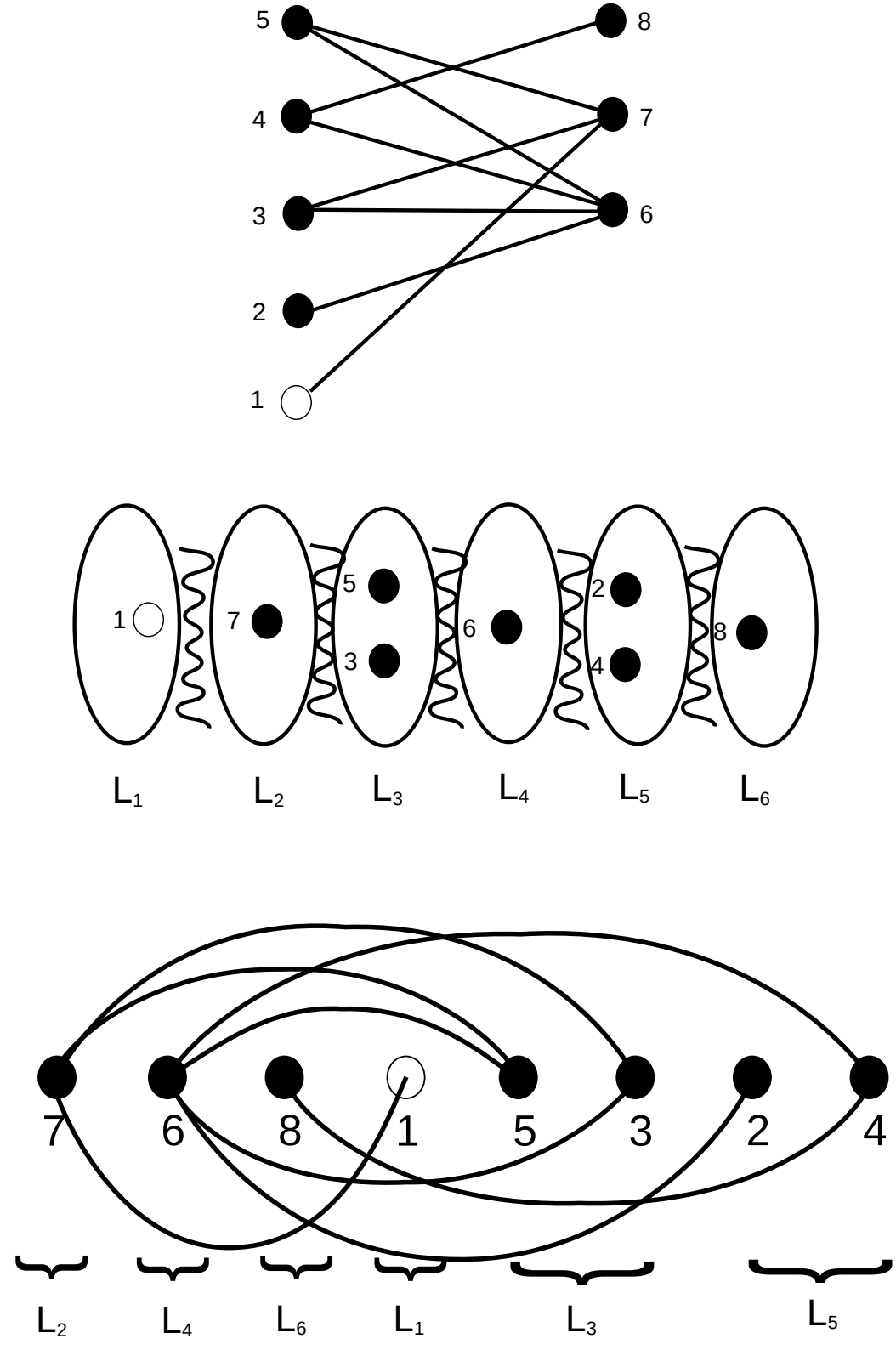

Figure 4.1: Trivial demonstration of proposed heuristic with seed vertex 1 producing antibandwidth of 3

shortest possible edge overrunning particular consecutive layers is greater by 1 than considered sum. If we consider bijection g^{-1} inverse to the one defined by formula (4.1), returning number of the layer placed on desired position (with assigned label resp.), and $d = d(D)$ we can state

$$ab(G) \geq ab(G, f) \geq \min_{k \in \left[2,\ \lceil d/2 \rceil + 1\right]} \left\{ 1 + \sum_{i=k}^{k+\lfloor d/2 \rfloor - 2} \left| L_{g^{-1}(i)} \right| \right\} \tag{4.4}$$

In the following, we will use $Hab(G)$ for the heuristical antibandwidth value – estimate produced by our method.

4.3 Parameters and improvements

The rough idea of heuristic was described in previous part, but there still are ways of affecting its result by parametrizing several steps. First is the choice of the seed, which can affect depth and minimum width of decomposition layers. Relevant question is which vertex or set of vertices to choose for the seed to produce optimal decomposition while still being efficient and not trying all possibilities and how big is the difference between sophisticated approach and random one. Another important decision affecting overall result that we discuss in this section is technique of labelling vertices inside layers. During the testing, which will be subject of following section 4.5, many configurations of these two parameters were tried. Although we cannot guarantee their best performance in general on all bipartite graphs and in special classes the changes can bring benefit, we rate following outcomes as best in average cases that we were able to obtain.

4.3.1 Seed selection

Seed as a independent set

Basic question is whether to choose a vertex or some independent set for a seed. Figure 4.2 shows, that choosing the suitable set could bring some

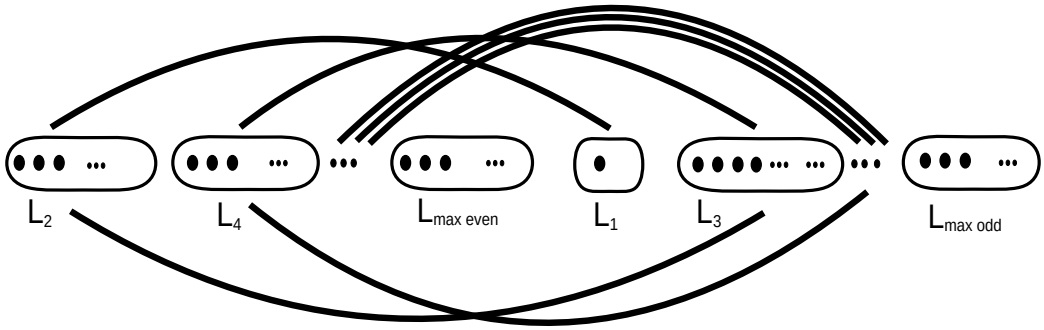

Figure 4.2: General image of ordering decomposition layers. Thick lines represent set of edges. Most of edges pass over L_1 and $L_{max\ even}$ layers

benefit because almost all edges in layout given by our heuristic goes over the seed set of vertices L_1 and over the set with the highest odd label L_k (assigned by formula (4.1)) and their width could increase length of most edges, making higher probability of better average result. We can hardly affect the size of L_k when decomposing the graph, but we can choose quite wide starting seed L_1. Strategy for choosing suitable independent set can be tough, because there are many of these and trying all of them while being efficient is impossible. It should be set with appropriate width and low "degree", meaning having not so wide neighbour set L_1, because we are still trying to maximize the depth of decomposition and it is natural that wider layers are, smaller the depth is. We can try to choose some proportionally adequate (relatively to number of all vertices) independent set of vertices with lowest degrees, but if the graph is dense enough, getting such set could not be easy in reasonable time. Another approach is to choose random vertex, construct decomposition and choose one of the first layers (because of relatively smaller "degree") that seems to be suitable for above conditions. During testing cases described in section 4.5 we tried these different strategies of choosing seed set, but results could not be evaluated as any big improvement in average, compared to proper choice of one vertex seed. For all of these reasons in following we focused only on case of single vertex seed.

Seed as a single vertex

If we choose only one vertex for a seed, minimum width of graph decomposition is 1, the worst possibility and according to formula (4.2) we have to only focus on increasing the depth of decomposition. It is possible to pass through all decompositions (starting in every vertex) and choose ones with biggest depth, order them within layers according to next section 4.3.2 and then from these take the one with highest antibandwidth as a result. We use this algorithm in the following. However, generating all possible decompositions takes $O(|V(G)|.|E(G)|)$ and if the order of G is big enough, we can come to problem with efficiency. It is natural and it was also observed that there is a correlation between vertex degree and decomposition depth. The lower degree initial vertex has, bigger depth of associated decomposition can be expected in average case. We can improve performance of heuristic by trying only decompositions induced by vertices of "low degree" same as it is done in Cuthill-McKee algorithm [4]. Or even better we can proceed like in Gibbs-Poole-Stockmeyer algorithm (GPS) [6] which is an improvement of original Cuthill-McKee and search for the endvertice of pseudo-diameter of the graph (defined by their algorithm) which is in average case seed for high depth decomposition. Since we are first interested in how appropriate results can our approach produce in general, we will use every vertex as a seed in order to get best possible antibandwidth estimate, but in the case of high order graphs and therefore also computational time, we can reduce the set of vertices used for a seed according to these algorithms.

4.3.2 Ordering within the layers

In this context we studied if some more sophisticated ordering of vertices within layers against random approach can mean significant difference in results. During testing the heuristic, we tried various different concepts, but the following one appeared to produce best averages. Let us have

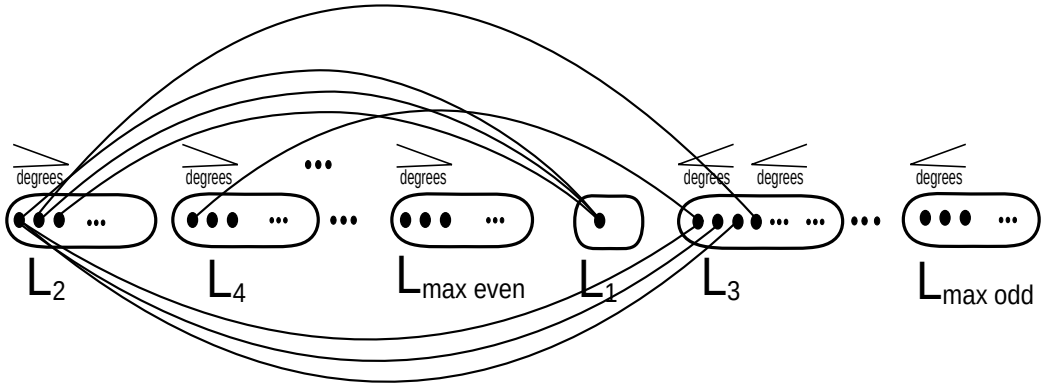

Figure 4.3: Demonstration of how the vertices are ordered within layers of decomposition

layer L_1 ordered (we let seed in natural order and in the case of one vertex thing is trivial). For each successive layer L_i, the vertices adjacent to the vertex in lowest assigned position in previous layer are consecutively placed in order of decreasing degree if the i is even and in the order of increasing degree if the i is odd. Ties being broken arbitrarily. Then unplaced vertices adjacent to the next vertex of lowest assigned position in previous layer are positioned in the same way. This process continues until all the vertices in the current layer are placed. Then the algorithm is applied to the next layer in the same manner. After all layers of decomposition are constructed in this way, we can order them according to formula (4.1) (with stable position of inside vertices) and finally now we can assign labels from 1 to $|V(G)|$ to placed vertices to obtain sought labelling f and compute lower bound of antibandwidth $ab(G, f)$. Simple demonstration of described process can be found on Figure 4.3.

Ordering neighbours in layers in sequential order relatively to their parents is apparently best way how to prolong edges connecting them with parents. Subordering the neighbours of the same vertex in decreasing resp. increasing order dependently on parity of layer should lower the probability of shorter edges causing worse result. If we look on Figure 4.3 we can see the reason. The vertices with high degrees should be placed as

far as possible from places that they are connected with, lowering the possibility of short edge. Even labelled layers are placed on the left relatively to they next odd neighbour layer and so the higher degree vertices within should be placed most to the left. On the other hand, odd labelled layers are to the right relatively to their neighbours and so high degree vertices should be placed most to the right as possible. In the Table 4.1 we can see some illustrative results for tree different types of ordering inside the layers. We are not stating them as statistical proof, but we could observe similar tendencies as presented during all later testing of heuristics and therefore in the following we will use described process of ordering.

Type of graph	**Described ordering within layers**	**Consecutive ordering without considering degree**	**Random ordering within layers**
$B_{500,0.5}$	12	10	5
$B_{500,0.5}$	12	11	5
$B_{500,0.5}$	12	12	5
$B_{500,0.5}$	12	10	5
$B_{500,0.5}$	14	11	5
$B_{1000,0.5}$	14	12	6
$B_{1000,0.5}$	13	12	6
$B_{1000,0.5}$	14	11	6
$B_{1000,0.5}$	14	13	5
$B_{1000,0.5}$	13	13	6
rT_{1000}	424	410	322
rT_{1000}	435	436	359
rT_{1000}	415	402	308
rT_{1000}	401	394	315
rT_{1000}	427	419	357

Table 4.1: Illustrative comparison of the antibandwidth heuristics with different ordering within layers. $B_{n,p}$ represents random bipartite graph with p edge probability and random separation of vertices to bipartitions one after another with probability $\frac{1}{2}$. rT_n represents random tree on n vertices

4.4 Connectivity

As we mentioned above, till this time we were considering only connected graphs. Let us now consider that

$$G = C_1 \cup C_2 \cup \ldots \cup C_m$$

is the union of components, with depth of decompositions

$$d_i = |D(C_i)|$$

and heuristical antibandwidth estimate

$$h_i = Hab(C_i)$$

for each component. During construction of final antibandwidth estimate, we can proceed in two directions outlined by constructive proofs of Theorems 3.1.3 and 3.1.4. First approach considers copies of the same graph, and works on a level of consecutive combining of single vertices from components in order to maximize overall edge length in average. We can generalize this method for iregular components by some kind of proportional merge of the component vertices based on cardinality of component and value of antibandwidth estimate. But since our components can largely differ in these values, proportional merging may not guarantee the optimal increase of local short edges.

On the other hand, we have advantage of linear decomposition for each component. We can make use of Theorem 3.1.4 for union of paths, and construct optimal layout for union of linear decompositions (optimal in sense of how many layers must the shortest possible edge overcome) in the same manner. Let overall decomposition of G consists of all consecutive layers from each component, L_{i,C_j} represents i-th layer of C_j. $D(G) = \{L_{1,C_1}, L_{2,C_1}, \ldots, L_{d_1,C_1}, L_{1,C_2}, L_{2,C_2}, \ldots, L_{d_m,C_m}\}$. This decomposition is "almost linear" – there are not connected neighbouring layers, if they come from different components, but all the rest of layers are connected in linear way. Now we can continue by applying the same procedure of assigning labels to layers by Formula (4.1) and constructing the

final linear layout. The depth of constructed decomposition is $d = \sum_{i=1}^{m} d_i$ and guarantees the shortest edge in the result to come over at least $\lfloor \frac{d}{2} \rfloor - 1$ layers.

4.5 Heuristic results and comparisons

In this section we are going to look on results that proposed heuristic produces on various bipartite graphs and their reasonability. Measurement of how the results are good or bad is still hard work, because as it was mentioned, there are very few exact or approximative results known, but if they are, most of them are for bipartite graphs and they are listed in Theorem 3.2.4. Proposed heuristic and graph structures were implemented in JAVA environment version SE 6 and run on average desktop machine. Tested classes were random trees, meshes, hypercubes, complete k-ary trees and arbitrary random bipartite graphs (but in this case there are no good results to compare with). For each class hundreds to thousands of different cases (depending on structure e.g hypercubes are exactly defined and regular) were subject of the heuristic and the results were stored as the ratio of heuristic result and the best known result in general (exact in some cases). Then the mean value of $\overline{RHab(G)}$ (Heuristical antibandwidth ratio) and the standard deviation σ is calculated as the evaluation of quality showing reasonability of heuristic in every case, giving image of how the heuristic could work on arbitrary bipartite graphs, where no exact results nor algorithms are known.

4.5.1 Random trees

Heuristic in this case was performed on random trees rT_n of different order. These were chosen quite high to prove its efficiency. As a comparison result (and best known for this time), lower bound given by Theorem 3.2.1 was used. As the order n of random tree goes high, it is natural that this

Order of rT_n	**Best known result**	**Heuristic result**	**Ratio**
1000	492	440	0.89
1000	488	426	0.87
1000	498	438	0.88
1000	490	434	0.89
1000	500	424	0.85
1000	498	433	0.87
1000	499	430	0.86
1000	489	421	0.86
1000	500	432	0.86
1000	500	437	0.87
1000	493	431	0.87
1000	493	424	0.86
1000	493	425	0.86
1000	498	427	0.86
1000	499	456	0.91
1000	486	434	0.89
1000	497	428	0.86
1000	493	427	0.87
⋮	⋮	⋮	⋮
		$\overline{RHab(rT_{1000})}$	0.87
		σ	0.02

Table 4.2: Results of the heuristic for random trees of order 1000

lower bound given by smaller bipartition of the tree tends to go to possible maximum of antibandwidth of $\lfloor \frac{n}{2} \rfloor$ (as can be seen in first column of table) and therefore can be considered optimal (expected random tree of big order has almost balanced bipartitions). Table 4.2 is just a short example of results, but mean value and standard deviation is calculated out of 1000 random samples.

When testing the heuristic on 1000 random trees of order 10000, statistical quality of results tends to go even higher to $\overline{RHab(rT_n)} = 0.89$ and $\sigma = 0.02$. Final statistical testing is trying to be universal in the sense, that

Order of rT_n	**Best known result**	**Heuristic result**	**Ratio**
13771	6855	6090	0.89
11073	5526	4841	0.88
787	387	348	0.9
6437	3207	2799	0.87
1925	953	844	0.89
4990	2475	2112	0.85
11178	5558	4903	0.88
16310	8123	7159	0.88
11232	5579	4924	0.88
18486	9222	8010	0.87
12342	6138	5601	0.91
18370	9135	8427	0.92
9015	4492	3941	0.88
2804	1389	1257	0.9
9027	4507	3919	0.87
9204	4579	4129	0.9
17838	8882	7891	0.89
7040	3510	3111	0.89
11628	5745	5116	0.89
⋮	⋮	⋮	⋮
		$\overline{RHab(rT_n)}$	0.88
		σ	0.02

Table 4.3: Results of the heuristic for random trees of random order up to 20000

orders of random trees are also generated randomly up to 20000 with results in Table 4.3. In this case of high order, not all the vertices are used as a seeds because of high computation time and decomposition is started only in chosen vertices of low degree as outlined in the end of section 4.3.1. Anyway the results are quite satisfying.

We can conclude that our heuristic is reasonable to use for trees, when it produces 0.88 of best known value of antibandwidth in average with quite narrow deviation without utilizing fact that graph is tree itself.

4.5.2 Mesh Graphs

Order of $M_{m\times n}$	**Exact result**	**Heuristic result**	**Ratio**
341×82	13940	13939	1.0
375×224	41888	41887	1.0
371×29	5365	5364	1.0
200×28	2786	2785	1.0
332×219	36245	36244	1.0
306×96	14640	14639	1.0
242×195	23498	23497	1.0
82×22	891	890	0.999
374×18	3357	3356	1.0
363×307	55567	55566	1.0
350×120	20940	20939	1.0
95×85	3995	3994	1.0
71×28	980	979	0.999
83×37	1517	1516	0.999
125×29	1798	1797	0.999
362×346	62453	62452	1.0
353×3	528	528	1.0
245×6	732	731	0.999
$\vdots$	$\vdots$	$\vdots$	$\vdots$
		$\overline{RHab(M_{m\times n})}$	0.999
		σ	0.002

Table 4.4: Results of the heuristic for mesh graphs of random order

Heuristic in this case was performed on mesh graphs $M_{m\times n}$, $m \geq n$ of different order generated randomly up to 400×400. These were also chosen quite high to prove if heuristic is efficient in cases, where no other general algorithm can be used. As a comparison result, exact value given by (iv) in Theorem 3.2.4 is used. Table 4.4 is just a short example of results, but mean value and standard deviation is calculated of 1000 random samples. We could see that heuristic produces almost optimal values, most of the time decreased just by 1 from the optimal one. Reason is that if we

have a deeper look on the proof of the optimal value, style of labelling the vertices is very similar to one that is made by our heuristic. But there are differences during ordering vertices within layers and that is the reason of this slight difference. From this result we can conclude, that our heuristic method is almost optimal for bipartite graphs of mesh type.

4.5.3 Hypercubes

Case of hypercubes Q_n is a bit special, because they are exact given regular structures with exponentially growing number of vertices and therefore not so many tests, which would prove statistical quality of the heuristic could be performed. Also the seed vertex can be chosen randomly, because of vertex symmetry of hypercubes. The highest hypercube that still could be modeled in heap memory of testing environment in a way necessary for heuristic algorithm was Q_{19} with 2^{19} vertices. As a reference result, asymptotically optimal lower bound (vi) from Theorem 3.2.4 enumerated by Maple 10 mathematical software (because of its complexity) is used. Comparison is in Table 4.5

We can see interesting result, because our heuristic in some cases even over performed best known theoretical result for hypercubes so far and in the rest ones is exactly equal. In the deeper look, our approach labels the vertices in very similar way that is used in the formal proof of (vi) in Theorem 3.2.4 in [13], but every $1(mod\ 4)$ case is special because of some simplification during proof. Although there are not many comparison results because of the style of labelling that our heuristic performs on hypercubes we can conclude that it produces asymptotically optimal results for hypercube graphs at least as good as best known lower bound.

4.5.4 Complete k-ary trees

Heuristic in this case was performed on complete k-ary trees of different even arity and height raising until the structure could be hold in the heap

Order of Q_n	**Best known result**	**Heuristic result**	**Ratio**
2^1	1	1	1.00
2^2	1	1	1.00
2^3	2	2	1.00
2^4	4	4	1.00
2^5	7	9	1.29
2^6	19	19	1.00
2^7	41	41	1.00
2^8	85	85	1.00
2^9	164	178	1.09
2^{10}	364	364	1.00
2^{11}	750	750	1.00
2^{12}	1522	1522	1.00
2^{13}	2976	3108	1.04
2^{14}	6280	6280	1.00
2^{15}	12756	12756	1.00
2^{16}	25708	25708	1.00
2^{17}	50611	52041	1.03
2^{18}	104707	104707	1.00
2^{19}	211469	211469	1.00

Table 4.5: Results of the heuristic for hypercubes Q_n

memory of the testing environment. Even arity is chosen, because in this case there is exact reference value given by (iii) of Theorem 3.2.4. Because amount of data is small (due to the structure and exponential growth of number of vertices), we omitted statistics.

This is the case, where we can see the weakness of our method. Wide hierarchical structure like complete trees of high arity means small decomposition depth. Because of such structure, cardinality of the consecutive layers is growing exponentially which leads to small number of very unproportional layers, starting with small ones and ending with huge layers relatively to the starting ones. That means most of the edges between wide layers to go over first relatively (to overall number of vertices) not so wide ones. In such cases our style of labelling vertices has very different philos-

Type of graph	**Best known result**	**Heuristic result**	**Ratio**
$T(4,2)$	9	4	0.44
$T(4,3)$	41	17	0.41
$T(4,4)$	169	68	0.4
$T(4,5)$	681	273	0.4
$T(4,6)$	2729	1092	0.4
$T(4,7)$	10921	4369	0.4
$T(4,8)$	43689	17476	0.4
$T(4,9)$	174761	69905	0.4
$T(6,2)$	19	6	0.32
$T(6,3)$	127	37	0.29
$T(6,4)$	775	222	0.29
$T(6,5)$	4663	1333	0.29
$T(6,6)$	27991	7998	0.29
$T(6,7)$	167959	47989	0.29
$T(8,2)$	33	8	0.24
$T(8,3)$	289	65	0.22
$T(8,4)$	2337	520	0.22
$T(8,5)$	18721	4161	0.22
$T(8,6)$	149793	33288	0.22
$T(10,2)$	51	10	0.2
$T(10,3)$	551	101	0.18
$T(10,4)$	5551	1010	0.18
$T(10,5)$	55551	10101	0.18
$T(12,2)$	73	12	0.16
$T(12,3)$	937	145	0.15
$T(12,4)$	11305	1740	0.15
$T(12,5)$	135721	20881	0.15

Table 4.6: Results of the heuristic for complete k-ary trees of height h T(k,h)

ophy than the optimal one used in a proof of exact value [16]. Result in this case cannot be considered satisfying and we can see that our style of heuristic is more convenient for kind of iregular graphs with more proportional decompositions in average like random trees or "product graphs" like meshes or hypercubes.

4.5.5 Random bipartite graphs

Type of graph	Heuristic result	Type of graph	Heuristic result
$B_{1000,0.05}$	94	$B_{1000,0.1}$	60
$B_{1000,0.05}$	93	$B_{1000,0.1}$	56
$B_{1000,0.05}$	94	$B_{1000,0.1}$	57
$B_{1000,0.05}$	94	$B_{1000,0.1}$	55
$B_{1000,0.05}$	96	$B_{1000,0.1}$	56
$B_{1000,0.05}$	96	$B_{1000,0.1}$	56
$B_{1000,0.05}$	95	$B_{1000,0.1}$	55
$B_{1000,0.05}$	94	$B_{1000,0.1}$	56
$B_{1000,0.05}$	96	$B_{1000,0.1}$	58
$B_{1000,0.05}$	94	$B_{1000,0.1}$	56
$B_{1000,0.05}$	98	$B_{1000,0.1}$	57
$B_{1000,0.05}$	95	$B_{1000,0.1}$	57
$B_{1000,0.05}$	100	$B_{1000,0.1}$	54
$B_{1000,0.05}$	95	$B_{1000,0.1}$	58
$B_{1000,0.05}$	98	$B_{1000,0.1}$	59
$B_{1000,0.05}$	94	$B_{1000,0.1}$	54
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\overline{Hab(B_{1000,0.05})}$	95.28	$\overline{Hab(B_{1000,0.1})}$	55.96
σ	2.02	σ	1.28

Table 4.7: Results of the heuristic for 200 randomly generated $B_{1000,0.05}$ and $B_{1000,0.1}$ bipartite graphs

Even though this class is in the centre of interest and we would like to know how appropriate our estimate of the antibandwidth is for the average random bipartite graph, we do not have any reference values to compare following results with and determine quality of our approach. Anyway, it can be interesting to see real values for real graphs. We chose to test our heuristic on graphs of type $B_{n,p}$ which represents random bipartite graph with p edge probability and random separation of n vertices to bipartitions one after another with probability $\frac{1}{2}$. It means graphs are likely to be balanced in average. Order of the graphs was chosen to 1000

and probability p was changed in the interval from 0.05 to 0.5. Results are summarized in following Tables 4.7, 4.8 and 4.9, statistics is made on the obtained values.

Type of graph	**Heuristic result**	**Type of graph**	**Heuristic result**
$B_{1000,0.2}$	33	$B_{1000,0.3}$	24
$B_{1000,0.2}$	33	$B_{1000,0.3}$	22
$B_{1000,0.2}$	33	$B_{1000,0.3}$	23
$B_{1000,0.2}$	33	$B_{1000,0.3}$	23
$B_{1000,0.2}$	32	$B_{1000,0.3}$	22
$B_{1000,0.2}$	33	$B_{1000,0.3}$	22
$B_{1000,0.2}$	32	$B_{1000,0.3}$	22
$B_{1000,0.2}$	32	$B_{1000,0.3}$	21
$B_{1000,0.2}$	31	$B_{1000,0.3}$	22
$B_{1000,0.2}$	33	$B_{1000,0.3}$	21
$B_{1000,0.2}$	33	$B_{1000,0.3}$	21
$B_{1000,0.2}$	32	$B_{1000,0.3}$	22
$B_{1000,0.2}$	32	$B_{1000,0.3}$	24
$B_{1000,0.2}$	33	$B_{1000,0.3}$	23
$B_{1000,0.2}$	34	$B_{1000,0.3}$	22
$B_{1000,0.2}$	32	$B_{1000,0.3}$	24
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\overline{Hab(B_{1000,0.2})}$	32.7	$\overline{Hab(B_{1000,0.3})}$	22.61
σ	0.85	σ	0.92

Table 4.8: Results of the heuristic for 200 randomly generated $B_{1000,0.2}$ and $B_{1000,0.3}$ bipartite graphs

Interesting fact can be that for each sample, values produced by the heuristic tend to flow round quite narrow interval. Explanation can be found in theory of random graphs. Key factor in our procedure is the depth of the decomposition. We are trying to maximize depth which is closely related to the graph diameter – in fact maximal decomposition depth is obviously greater by 1 than the graph diameter. Summarized results on diameters of random bipartite graphs can be found in Bollobas'

Type of graph	**Heuristic result**	**Type of graph**	**Heuristic result**
$B_{1000,0.4}$	16	$B_{1000,0.5}$	14
$B_{1000,0.4}$	17	$B_{1000,0.5}$	15
$B_{1000,0.4}$	16	$B_{1000,0.5}$	13
$B_{1000,0.4}$	18	$B_{1000,0.5}$	13
$B_{1000,0.4}$	17	$B_{1000,0.5}$	13
$B_{1000,0.4}$	16	$B_{1000,0.5}$	12
$B_{1000,0.4}$	16	$B_{1000,0.5}$	13
$B_{1000,0.4}$	16	$B_{1000,0.5}$	15
$B_{1000,0.4}$	17	$B_{1000,0.5}$	12
$B_{1000,0.4}$	17	$B_{1000,0.5}$	14
$B_{1000,0.4}$	15	$B_{1000,0.5}$	12
$B_{1000,0.4}$	19	$B_{1000,0.5}$	13
$B_{1000,0.4}$	16	$B_{1000,0.5}$	13
$B_{1000,0.4}$	16	$B_{1000,0.5}$	12
$B_{1000,0.4}$	17	$B_{1000,0.5}$	14
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\overline{Hab(B_{1000,0.4})}$	17.23	$\overline{Hab(B_{1000,0.5})}$	13.32
σ	0.74	σ	0.57

Table 4.9: Results of the heuristic for 200 randomly generated $B_{1000,0.4}$ and $B_{1000,0.5}$ bipartite graphs

book [1]. Key fact is that for balanced bipartitions and given p, there always is an estimate for expected diameter. For example the case of balanced bipartitions and $p = \frac{1}{2}$ is proved to have expected diameter of 3, which corresponds with the typical number of 4-layer decomposition that we observed in this case. When the decomposition depth is almost stable for each case and the edges are distributed randomly between layers, it comes natural that obtained heuristic values are as similar as presented. The estimate for real expected antibandwidth of random bipartite graphs would be the subject of another research.

4.6 Heuristic complexity

Let us consider full heuristic with decomposition starting in every vertex of G, $|V| = |V(G)|$ and $|E| = |E(G)|$. Every of these decompositions requires to traverse all the edges to construct it. During this process there is a local sorting on neigbours done, which can be upper bounded by sorting of all vertices (which is the worst case that can happen at the same time, when e.g. the seed vertex is neigbour of all other vertices and they have to be sorted). At the end, for each of the decompositions, evaluation of antibandwidth estimate is done, which requires traversing all edges of the given graph. From this analysis, we can conclude that complexity of our heuristic is $O\Big(|V|.(|E|+|V|.log|V|+|E|)\Big)$. In the worst but most usual case of $|E| \sim |V|^2$ mean that our heuristic runs in $O(|V|^3)$. Improvement can be reached by sophisticated and not exhaustive choices of the seed vertices as outlined in section 4.3.1 and recording the actual antibandwidth during process.

Chapter 5

Conclusions

Antibandwidth problem is one of the interesting NP-complete labelling resp. linear layout problems on graphs. We showed the background of the problem, known theoretical results and reasons why it is hard to be obtained. After we proposed one kind of heuristic for producing lower bound estimate for antibandwidth parameter of bipartite graphs, because no approximation procedure for the problem is known so far. Even though the testing the quality of heuristic is problem because of non-availability of many reference results, for existing ones our procedure performed quite well in most comparable cases and we discussed the convenience of use for every particular tested class. In the case of random bipartite graphs, where there are no comparison data available, we can just offer our results as the best general estimate of antibandwidth parameter which is known.

There is still a lot of work on this field, heuristic could be improved to perform better also for hierarchical structures like complete k-ary trees, complexity could be decreased by more sophisticated choices of seed and modification of layers of decomposition. Results could be compared to some standard general global optimization procedures like *simulated annealing* or *genetic algorithms*. But the major challenge remains idea of general heuristic for the antibandwidth problem.

Bibliography

[1] Bollobás, B., Random Graphs, Cambridge University Press, 2001, ISBN 0521797225

[2] Cappanera, P., A Survey on Obnoxious Facility Location Problems, citeseer.ist.psu.edu/cappanera99survey.html, (1999)

[3] Chinn, P.Z., Chvatalova, J., Dewdney, A.K, Gibbs, N., E., The Bandwidth Problem for Graphs and Matrices – A Survey, Journal of Graph Theory, Vol. 6 (1982), 223-254

[4] Cuthill, E., McKee, J., Reducing the bandwidth of sparse symmetric matrices, Proc. 24th Nat. Conf. ACM (1969), 157-172

[5] Donnelly, S., Isaak, G., Hamiltonian powers in treshold and arborescent comparability graphs, Discrete Mathematics 202 (1999), 33-44

[6] Gibbs, N.E, Poole Jr., W.G., Stockmeyer, P.K, An alghoritm for reducing the bandwidth and profile of sparxe matrix, SIAM J.Numer. Anal. 13 (1976) 235-251

[7] Göbel, F., The separation number, Ars Combinatoria 37, (1994), 262-274

[8] Hale, W. K., Frequency assignment: theory and applications, Proceedings of IEEE 60 (1980), 1497-1514

[9] Hassin R., Rubinstein, S., Approximation algorithms for maximum linear arrangement, Information Processing Letters 80 (2001), 171-177

[10] Isaak, G., Powers of Hamiltonian Paths in Interval Graphs, Journal of Graph Theory 27 (1998), 31-38

[11] Leung, J.Y-T., Vornberger, O., Witthoff, J.D, On some variants of the bandwidth minimization problem, SIAM J. Computing 13 (1984), 650-667

[12] Lin, Y., Yuan, J.J., The dual bandwidth problem for graphs, J. Zhengzhou Uni. Nat. Sci. Ed. 35 (2003), 1-5

[13] Miller, Z., Pritikin, D., On the separation number of a graph, NETWORKS 19 (1989), 651-666

[14] Miller, Z., Pritikin, D., Eigenvalues and Separation in Graphs, Linear Algebra and its Applications 181 (1993), 187-219

[15] Raspaud, A., Schröder, H., Sýkora, O., Török, L., Vrťo, I., Antibandwidth and Cyclic Antibandwidth of Meshes and Hypercubes, submitted to Discrete Mathematics

[16] Török, L., Two problems in graph layouts, PhD. thesis, Slovak Academy of Science, 2007

Abstrakt

Problém *antibandwidth* pre graf $G = (V, E)$ pozostáva z označenia jeho $|V(G)| = n$ vrcholov v_i rôznymi prirodzenými číslami $f(v_i)$ v rozsahu $[1, n]$ ($f : V \to \{1, 2, \ldots n\}$ je bijektívne zobrazenie) takým spôsobom, že hodnota výrazu

$$min\{|f(v_i) - f(v_j)| : (v_i v_j) \in E(G)\}$$

je maximalizovaná, ak uvažujeme všetky možné zobrazenia f. Túto hodnotu potom označujeme ako *antibandwidth* G. Niekedy býva tento problém formulovaný aj ako problém lineárneho rozmiestnenia grafu. Ak sa na hodnoty $f(v_i)$ pozrieme ako na body umiestnenia vrcholu v_i na číselnú os v rozsahu $[1, n]$ (toto lineárne rozmiestnenie grafu priradí každej jeho hrane presnú dĺžku) tak hodnotu *antibandwidth* určuje práve také lineárne rozmiestnenie pre graf G, kde dĺžka najkratšej hrany je maximalizovaná a táto hodnota je práve hodnotou parametra. Tento problém sa pôvodne objavil ako duálna variácia oveľa známejšieho problému *bandwidth*, ale dnes nachádza mnohé iné interpretácie - ako problém pre multiprocesorové rozvrhovanie úloh alebo problém umiestňovania neznášavých entít. Problém *antibandwidth* je NP-úplný a existuje len veľmi málo presných výsledkov pre netriviálne triedy grafov a niektoré triedy, kde je výpočet časovo polynomiálny. Zatiaľ nie je známa žiadna všeobecná heuristika použiteľná pre tento problém. Pre triedu bipartitných grafov ani nie je známe, či je problém tiež ťažký alebo existuje časovo polynomiálny algoritmus pre jeho riešenie. V tejto práci, po zhrnutí všetkých základných známych výsledkov o probléme, predstavíme jednu verziu heuristiky získavajúcu konštrukčný dolný odhad pre ľubovolné bipartitné grafy. Popíšeme jej variácie a ich vplyv na kvalitu výsledku a zložitosť algoritmu. Potom predstavíme štatistické porovnanie výsledkov navrhnutej heuristiky k doteraz známym faktom a zhodnotíme jej použiteľnosť pre jednotlivé triedy grafov.

Kľúčové slová: teória grafov, antibandwidth, dual bandwidth