

# Multilevel Algorithms for Linear Ordering Problems.

Ilya Safro\*

Dorit Ron

Achi Brandt

July 27, 2005

## Abstract

Linear ordering problems are combinatorial optimization problems which deal with the minimization of different functionals in which the graph vertices are mapped onto  $(1, 2, \dots, n)$ . These problems are widely used and studied in many practical and theoretical applications. In this paper we present a variety of linear-time algorithms for these problems inspired by the Algebraic Multigrid approach which is based on weighted edge contraction. The experimental results for four such problems turned out to be better than every known results in almost all cases, while the short running time of the algorithms enables testing very large graphs.

## 1 Introduction

The objective of the class of linear ordering problems is to minimize different functionals that map the set of the graph vertices onto  $(1, 2, \dots, n)$ . This class contains many graph (or matrix) layout problems such as : the minimum  $p$ -sum, the workbound reduction, the wavefront, the envelope size, etc. Some problems, such as finding the minimum linear arrangement [28] or the bandwidth [22], appear in many applications for solving problems in the large sparse matrix computation. Some other are closely related to the problem of calculating the envelope size of a symmetric matrix or, more precisely, to the amount of work needed in the Cholesky factorization of such a matrix [16]. Linear ordering problems may also be motivated as a model used in VLSI design [11] and may be used in several biological applications, graph drawing and other fields (see [14, 22, 17, 31]). Commonly for general graphs (or matrices) these problems are NP-hard and their decision versions are NP-complete [15].

Since these problems have a practical significance, many heuristic algorithms were developed in order to achieve near optimal solution. Among the most successful are Spectral Sequencing [19, 12], Optimally Oriented Decomposition Tree [1], Multilevel based [21, 18],

---

\*Corresponding author : ilya.safro@weizmann.ac.il

Simulated Annealing [24], Genetic Hillclimbing [25] and others. Some of these algorithms have proven themselves superior in solution quality while others in execution time.

In this paper we present a general framework of multilevel algorithms especially designed for linear ordering problems. Our strategy is based on the Algebraic MultiGrid scheme (AMG) [3, 4, 5, 10, 27, 33, 34]. While in previous works we have developed and tested special multilevel algorithms for solving the minimum linear arrangement problem [28] and the minimum 2-sum problem [29], in this article we demonstrate how the building blocks of the general multilevel approach can be used in various ways to make it suitable for solving more involved functionals. In particular, we present two algorithms : we show how the *bandwidth* of a graph can be approximated by a continuation approach in which a sequence of increasingly  $p$ -sum problems are involved until  $p$  is large enough to be considered infinite for practical purposes; in addition, we use the minimum 2-sum problem result as a first approximation for the *workbound* reduction problem, which is then improved by a postprocessing of local minimizations with actual use of the workbound functional. In fact, we propose to use the ordering obtained by the minimum 2-sum problem as a first approximation for other linear ordering problems, as demonstrated for the *wavefront reduction* problem.

The main objective of a multilevel based algorithm is to create a hierarchy of problems, each representing the original problem, but with fewer degrees of freedom. General multilevel techniques have been successfully applied to various areas of science (e.g. physics, chemistry, engineering, etc.) [7, 9]. AMG methods were originally developed for solving linear systems of equations resulting from the discretization of partial differential equations. Lately they have been applied to various other fields, yielding for example novel methods for image segmentation [32] and for the linear arrangement problem [28]. In the context of graphs it is the Laplacian matrix that represents the related set of equations. The main difference between our approach to most other multilevel approaches (related to various graph optimization problems) is the coarsening scheme. While the previous approaches may be viewed as *strict* aggregation process, the AMG coarsening is actually a *weighted* aggregation : each node may be divided into *fractions*, and different fractions belong to different aggregates. This enables more freedom in solving the coarser levels and avoids making hardened local decisions, such as edge contractions, before accumulating the relevant global information.

It turned out that this general coarsening is suitable for all the different functionals we have tested. The various algorithms thus differ in the disaggregation process which follows by projecting to a finer level the final arrangement obtained on a coarser level. This initial fine level arrangement is being further improved by applying different local reordering methods. In addition to node by node minimization, we have developed a simultaneous minimization of several vertices called Window Minimization. In its basic application (for the 2-sum problem [29]) it involves the minimization of a quadratic form. Here we show how to quadratize other functionals. Also, we suggest the use of numerical calculation rather than analytic, for instance, in calculating derivatives. Finally, our postprocessing is intensified by Simulated Annealing (SA) [20] which is a general method to escape local minima. In the multilevel framework SA is aimed at searching only for *local* changes that guarantee the preservation of large-scale solution features inherited from coarser levels.

We compared the results obtained by our multilevel algorithms with many previously described algorithms. In this paper we present the results of the bandwidth problem and

the workbound problem and show that our results are on the average better than previous ones by about 30%. Our experimental results show that the AMG framework can be used for linear ordering problems to obtain high quality results in linear time.

The various functionals and their generalizations are described in Sec. 2. The multi-level algorithm along with additional optimization techniques are presented in Sec. 3. A comparison of our results with other works is finally summarized in Sec. 4.

## 2 Definitions and generalizations

Given a weighted graph  $G = (V, E)$ , where  $V = \{1, 2, \dots, n\}$ , denote by  $w_{ij}$  the non-negative weight of the edge  $ij$  between nodes  $i$  and  $j$ ; if  $ij \notin E$  then  $w_{ij} = 0$ . Let  $\pi$  be a bijection

$$\pi : V \longrightarrow (1, 2, \dots, n) \quad .$$

The purpose of linear ordering problems is to minimize some functional over all possible permutations  $\pi$ . The following functional should be minimized for the minimum  $p$ -sum problem<sup>1</sup> :

$$\sigma_p(G, \pi) = \sum_{ij} w_{ij} |\pi(i) - \pi(j)|^p \quad . \quad (1)$$

In the generalized form of the problem that emerges during the multilevel solver, each vertex  $i$  is assigned with a *volume* (or *length*), denoted  $v_i$ . The task now is to minimize the cost  $\sigma_p(G, x) = \sum_{ij} w_{ij} |x_i - x_j|^p$ , where  $x_i = \frac{v_i}{2} + \sum_{k, \pi(k) < \pi(i)} v_k$ , i.e., each vertex is positioned at its center of mass capturing a segment on the real axis which equals its length. The original form of the problem is the special case where all the volumes are equal. In particular, we would like to concentrate on the minimum bandwidth problem which seeks a linear layout that minimizes the maximal stretched edge, i.e.,  $bw(G) = \min_{\pi} \max_{ij} w_{ij} |\pi(i) - \pi(j)|$ . The minimization functional of the bandwidth problem can be formulated in term of  $\sigma_p(G, \pi)$  :

$$bw(G, \pi) = \lim_{p \rightarrow \infty} (\sigma_p(G, \pi))^{1/p} \quad . \quad (2)$$

The minimization functional of the workbound reduction problem is defined as

$$wb(G, \pi) = \sum_i \max_{\substack{j \\ \pi(j) < \pi(i)}} w_{ij} (\pi(i) - \pi(j))^2 \quad . \quad (3)$$

The generalized form of this problem is similar to the above derivation, and the max function may be approximated by

$$wb(G, x) = \sum_i \max_{j: x_j < x_i} w_{ij} (x_i - x_j)^2 \approx \sum_i \left( \sum_{j: x_j < x_i} w_{ij} (x_i - x_j)^p \right)^{2/p} \quad . \quad (4)$$

We will not discuss here theoretical complexity issues, such as lower and upper bounds for the solution cost. We are not interested in the worst possible cases, which are extremely

---

<sup>1</sup>We use this definition for simplicity, while the usual definition of the functional is  $\sigma_p(G, \pi) = (\sum_{ij} w_{ij} |\pi(i) - \pi(j)|^p)^{1/p}$ , which yields of course the same minimization problem.

non-representative. Our focus is on practical high-performance algorithms, such that in most practical cases would yield a good approximation to the optimum at low computational cost. Typically, the multilevel algorithms exhibit linear complexity, i.e., the computational cost in most practical cases is proportional to  $|V| + |E|$ .

### 3 The algorithm

In the multilevel framework a hierarchy of decreasing size graphs :  $G_0, G_1, \dots, G_k$  is constructed. Starting from the given graph,  $G_0 = G$ , create by recursive *coarsening* the sequence  $G_1, \dots, G_k$ , then solve the coarsest level  $G_k$  directly, and finally uncoarsen the solution back to  $G$ . This entire process is called a *V-cycle*. As in the general AMG setting, the choice of the coarse variables (aggregates), the derivation of the coarse problem which approximates the fine one and the design of the coarse-to-fine disaggregation (uncoarsening) process are all determined automatically, as described below.

#### 3.1 Coarsening: Weighted Aggregation

The coarsening used here is similar to the process we have used in solving the minimum linear arrangement and the minimum 2-sum problems [28, 29]. For completeness we briefly repeat its description. The coarsening is interpreted as a process of *weighted aggregation* of the graph nodes to define the nodes of the next coarser graph. In a *strict* aggregation process (also called edge contraction or matching of vertices) the nodes are blocked in small disjoint subsets, called aggregates. Two nodes  $i$  and  $j$  would usually be blocked together (put in the same aggregate) if their coupling is *strong*, meaning that  $w_{ij}$  is comparable to  $\min\{\max_k w_{ik}, \max_k w_{kj}\}$ . In *weighted* aggregation, each node can be divided into *fractions*, and different fractions belong to different aggregates. In both cases, these aggregates will form the nodes of the *coarser level*, where they will be blocked into larger aggregates, forming the nodes of a *still coarser level*, and so on. As AMG solvers have shown, *weighted*, instead of *strict*, aggregation is important in order to express the *likelihood* of nodes to belong together; these likelihoods will then accumulate at the coarser levels of the process, automatically reinforcing each other where appropriate. Strict aggregation, by contrast, may run into a conflict between the local blocking decision and the larger-scale picture.

The construction of a coarse graph from a given one is divided into three stages: first a subset of the fine nodes is chosen to serve as the *seeds* of the aggregates (the later being the nodes of the coarse graph), then the rules for interpolation are determined, thereby establishing the fraction of each non-seed node belonging to each aggregate, and finally the strength (or weight) of the connections (or edges) between the coarse nodes is calculated.

We propose the following coarsening scheme for all kinds of linear ordering problems. The same common procedure was used for the minimum  $p$ -sum problem as well as for the workload problem.

**Coarse Nodes.** The construction of the set of seeds  $C$  and its complement, denoted by  $F$ , is guided by the principle that each  $F$ -node should be “strongly coupled” to  $C$ . Also, we will include in  $C$  nodes with exceptionally large volume, or nodes expected (if used as seeds) to aggregate around them exceptionally large volumes of  $F$ -nodes. To achieve these

objectives, we start with an empty set  $C$ , hence  $F = V$ , and then sequentially transfer nodes from  $F$  to  $C$  until all remaining  $i \in F$  satisfy

$$\sum_{j \in C} w_{ij} / \sum_{j \in V} w_{ij} \geq Q \quad ,$$

where  $Q$  is a parameter;  $Q = 0.4$  is used in the reported experiments.

**The Coarse Problem.** Each node in the chosen set  $C$  becomes the seed of an aggregate that will constitute one coarse level node. Define for each  $i \in F$  a coarse neighborhood  $N_i = \{j \in C, w_{ij} \geq \alpha_i\}$ , where  $\alpha_i$  is determined by the demand that  $|N_i|$  does not exceed the allowed coarse neighborhood size  $r$  chosen to control complexity (typical values of  $r$  are given in the Appendix), while neighbors with larger  $w_{ij}$  are chosen first and those with much smaller  $w_{ij}$  (say 100 times smaller than the maximal one) are neglected. Let  $I(j)$  be the ordinal number in the coarse graph of the node that represents the aggregate around a seed whose ordinal number at the fine level is  $j$ . The classical AMG interpolation matrix  $P$  (of size  $|V| \times |C|$ ) is defined by

$$P_{iI(j)} = \begin{cases} w_{ij} / \sum_{k \in N_i} w_{ik} & \text{for } i \in F, j \in N_i \\ 1 & \text{for } i \in C, j = i \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

$P_{iI}$  represents the likelihood of  $i$  to belong to the  $I$ -th aggregate. Following the weighted aggregation scheme used in [32], the edge connecting two coarse aggregates  $p$  and  $q$  is assigned with the weight  $w_{pq} = \sum_{k \neq l} P_{kp} w_{kl} P_{lq}$ . The volume of the  $p$ -th coarse aggregate is  $\sum_j v_j P_{jp}$ . Note that the process of coarsening conserves the total volume of all vertices.

### 3.2 The coarsest level

Minimizing the appropriate functional at the coarsest level, which consists of no more than 8 nodes (otherwise a still coarser level would be introduced for efficiency) is performed directly by simply trying all possible arrangements. Since the amount of work invested at the coarsest levels is small compared with that of the finest level, many solutions can in fact be kept at each level whose graph is small relative to  $G$ . In principle, this number depends on the amount of work associated with the graph parameters of that level. In particular, a large number of solutions is chosen at the coarsest level; they are chosen so that they all enjoy a relatively low energy cost and are mutually significantly different from each other. Each is then propagated to the next finer level and being optimized there. The best solutions are chosen using the same criteria, and so on. This variety of solutions enables a better sampling of the space of permutations without much affecting the efficiency of the algorithm.

Since we wanted to measure the standard deviation for our algorithm, we have run it a few times for each of the given graphs by starting with a different permutation of the nodes of  $G$  (see Section 4.2). Experiments show that it became less important to also have a multitude of solutions at each level. Still this approach has proven to work well for [26].

### 3.3 Disaggregation (uncoarsening)

While the same identical coarsening procedure was used for the minimization of all our functionals, the uncoarsening only shares the same basic structure, but the actual implementation varies from one functional to another. Having solved a coarse problem, the solution to the next-finer-level problem is initialized by first placing the seeds according to the coarse order and then adjusting all other  $F$ -nodes while aiming at the minimization of the arrangement cost. This first approximation is subsequently improved by several *relaxation* sweeps, first compatible, then regular (explained below). Then, the arrangement is improved by strict minimization, possibly with added stochasticity. These are the local reordering processes which either accept only changes that decrease the arrangement cost (strict minimization) or might also accept steps which increase the cost (with some probability) in order to escape false local minima (simulated annealing). The entire scheme is explained below and summarized in Algorithm 2.

Before we turn to the details of these common stages of the disaggregation process, let us describe the particular structure we have used for the minimum  $p$ -sum problem. The disaggregation scheme for the minimization of  $\sigma_p(G, x)$  is based on *continuation* in the parameter  $p$ , such that  $p = 2$  is used to exactly solve the coarsest level, and then, at each subsequent finer level,  $p$  is increased (e.g. by two). Thus, every level  $l$  (other than the coarsest) minimizes  $\sigma_p(G_l, x)$  by initialization from  $\sigma_{p-2}(G_{l+1}, x)$ . Except that in cases where the desired  $p$  is already reached on one of the coarse levels, no further continuation is employed beyond that level. Our experiments show that the results are not sensitive to small changes in the continuation of  $p$ , e.g., solving the coarsest level with  $p = 4$ , or increasing  $p$  by four. In case where  $p$  should tend to infinity (as for the bandwidth (2)), the increase of  $p$  is continued also at the end of the V-cycle in a *postprocessing* procedure.

#### 3.3.1 Initialization of the next finer level

Given is the arrangement of the coarse level aggregates in its generalized form, where the center of mass of each aggregate  $j \in C$  is positioned at  $x_{I(j)}$  along the real axis. We begin the initialization of the fine level arrangement by letting each seed  $j \in C$  inherit the position of its respective aggregate:  $y_j = x_{I(j)}$ . At each stage of the initialization procedure, define  $V' \subset V$  to be the subset of nodes that have already been placed, so we start with  $V' = C$ . Then proceed by positioning each fine node  $i \in V \setminus V'$  at the coordinate  $y_i$  in which the cost of the arrangement, at that moment when  $i$  is being placed, is minimized. The sequence in which the nodes are placed is roughly in decreasing order of their *relative* connection to  $V'$ , that is, the nodes which have strong connections to  $V'$  compared with their connections to  $V$  are placed first. To be precise, for the minimum  $p$ -sum problem the coordinate  $y_i$  is located at its minimum (volumes are not taken into account)

- if  $p = 1$  then  $y_i \in \{y : |\sum_{y_j < y, j \in V'} w_{ij} - \sum_{y_j > y, j \in V'} w_{ij}| \text{ is minimal}\}$ , i.e.,  $y_i$  is within the median segment,
- if  $p = 2$  then  $y_i = \frac{\sum_{j \in V'} y_j w_{ij}}{\sum_{j \in V'} w_{ij}}$ , i.e.,  $y_i$  is placed at the weighted average position of  $y_j$ ,  $j \in V'$ , to which  $y_i$  is connected,

- for a general (even)  $p$  the location of  $y_i$  has to minimize  $\sum_{j \in V'} w_{ij}(y_i - y_j)^p$ . This is achieved numerically by several steps of Newton-Rhapson method starting at the  $p = 2$  solution.

Then  $V' \leftarrow V' \cup \{i\}$  and the process continues until  $V' = V$ . Finally each position  $y_i$  is changed to

$$x_i = \frac{v_i}{2} + \sum_{y_k < y_i} v_k \quad , \quad (6)$$

thus retaining order while taking volume (length) into account.

### 3.3.2 Relaxation

The arrangement obtained after the initialization is a first feasible solution for the minimum  $p$ -sum problem which is then improved by employing several sweeps of *relaxation*, first *compatible* then *Gauss-Seidel-like*. These two types of relaxation are very similar to the above initialization: The compatible relaxation, motivated in [8], improves the positions of the  $F$ -nodes one by one according to the minimization criteria above (where  $V' = V$ ) while keeping the positions of the seeds ( $C$ -nodes) unchanged. The Gauss-Seidel-like relaxation is similarly performed, but for *all* nodes (including  $C$ ). Each such sweep is again followed by (6).

### 3.3.3 Windows Minimization

The cost of the arrangement can be further reduced by *strict minimization*, i.e., a sequence of rearrangement that accepts only changes which decrease the arrangement cost. Since done in the multilevel framework, it can be restricted at each level to just *local* changes, i.e., reordering small sets of neighboring nodes, which are adjacent (or relatively close) to each other at the current arrangement. It is easy to see that switching positions between several adjacent nodes is inexpensive, since the resulting new arrangement cost can be calculated only at the vicinity of the adjustment and not elsewhere. Such a node by node minimization was applied in our algorithm for the Minimum Linear Arrangement problem (1-sum problem, see [28]). This method may also be used for any functional. However, for the minimum 2-sum problem we have introduced a more advanced method of local minimization, called *Windows Minimization* (WM), which is suitable not only for the multilevel framework but can also be used as local postprocessing relaxation in other frameworks (like the spectral approach). The difference between WM and simple node by node minimization is that WM *simultaneously* minimizes the arrangement cost of a small number of nodes (e.g., 5 to 20).

We first describe the basic WM involving the quadratic form for  $p = 2$  [29], then possible generalizations are presented. Given a current approximation  $\tilde{x}$  to the arrangement of the graph, denote by  $\delta_i$  a *correction* to  $\tilde{x}_i$ . Let  $\mathfrak{W} = \{i_1 = \pi^{-1}(s+1), \dots, i_q = \pi^{-1}(s+q)\}$  be a *window*, i.e.,  $q$  successive vertices in the current arrangement, positioned at  $\tilde{x}_{i_1}, \dots, \tilde{x}_{i_q}$ . The local minimization problem of the  $p = 2$  functional associated with a given window  $\mathfrak{W}$  can be formulated as follows :

$$\text{minimize } \sigma_2(\mathfrak{W}, \tilde{x}, \delta) = \sum_{i,j \in \mathfrak{W}} w_{ij}(\tilde{x}_i + \delta_i - \tilde{x}_j - \delta_j)^2 + \sum_{\substack{i \in \mathfrak{W} \\ j \notin \mathfrak{W}}} w_{ij}(\tilde{x}_i + \delta_i - \tilde{x}_j)^2. \quad (7)$$

To prevent the possible convergence of many coordinates to one point, and, more precisely, to express the aim of having  $\{x_i + \delta_i\}_{i \in \mathfrak{W}}$  an approximate permutation of  $\{x_i\}_{i \in \mathfrak{W}}$  one should add constraints of the form

$$\sum_{i \in \mathfrak{W}} (\tilde{x}_i + \delta_i)^m v_i = \sum_{i \in \mathfrak{W}} \tilde{x}_i^m v_i . \quad (8)$$

For simplicity, we have used only the first two moments, where for  $m = 2$  we have neglected the quadratic term in  $\delta_i$ . Using Lagrange multipliers, the final formulation of the WM for  $p = 2$  is :

$$\text{minimize } \sigma_2(\mathfrak{W}, \tilde{x}, \delta, \lambda_1, \lambda_2) = \sigma_2(\mathfrak{W}, \tilde{x}, \delta) + \lambda_1 \sum_{i \in \mathfrak{W}} \delta_i v_i + \lambda_2 \sum_{i \in \mathfrak{W}} \delta_i v_i \tilde{x}_i , \quad (9)$$

under the second and third constraints of (10) below, yielding the following system of equations:

$$\begin{cases} \sum_{j \in \mathfrak{W}} w_{ij} (\delta_i - \delta_j) + \delta_i \sum_{j \notin \mathfrak{W}} w_{ij} + \lambda_1 v_i + \lambda_2 v_i \tilde{x}_i = \sum_j w_{ij} (\tilde{x}_j - \tilde{x}_i) & \text{for } i = 1, \dots, q \\ \sum_i \delta_i v_i = 0 \\ \sum_i \delta_i v_i \tilde{x}_i = 0 \end{cases} . \quad (10)$$

Usually in a correct multilevel framework, the changes  $\delta_i$  are supposed to be relatively small since the global approximation for the arrangement is inherited from the coarser levels. Their smallness is effected by the very restriction of the minimization to one window at a time. After solving the system (10), every vertex  $i \in \mathfrak{W}$  is thus positioned at  $y_i = \tilde{x}_i + \delta_i$ . Feasibility with respect to the volumes of the nodes is retained by applying (6). Since the size and location of  $\mathfrak{W}$  are quiet arbitrary, the energy cost of the new sub-arrangement can be further improved by Gauss-Seidel-like relaxation sweeps applied to an *enlarged* window  $\mathfrak{W}$ , where, say 5% of the window's size at each end (if possible) are added to  $\mathfrak{W}$ . As usual, each sweep is followed by (6). The final obtained energy cost is then compared with the one prior to all the window changes, the minimum of the two is accepted, updating  $\tilde{x}$ .

A sweep of WM with a given window size  $q$  consists of a sequence of overlapping windows, starting from the first node in the current arrangement and stepping by  $\lfloor \frac{q}{2} \rfloor$  for each additional window. One such sweep is employed for every given  $q$ , while a small number of different  $q$ 's is used (for actual values see Sections 4.2 and 4.3). Our experiments show that the algorithm is robust to changes in the chosen  $q$ 's. Note that due to the multiscale framework, only bounded values of  $q$  need be used, which guarantees linear execution time of the entire algorithm. The WM is summarized in Algorithm 1.

**Algorithm 1:** WindowsMinimization(graph  $G$ , current order  $\tilde{x}$ , window length  $q$ )

**Parameter:**  $k_1$

**For**  $i = 1$  **To**  $|V| - q + 1$  **Step**  $i = i + \lfloor \frac{q}{2} \rfloor$

$\mathfrak{W} = \{\pi^{-1}(i), \dots, \pi^{-1}(i + q - 1)\}$

**Solve** the system of equations (10)

**Apply**  $k_1$  sweeps of Gauss-Seidel-like relaxation on the enlarged  $\mathfrak{W}$  with  $\tilde{x} + \delta$

$\tilde{x} \leftarrow \tilde{x} + \delta$  if the cost of the arrangement was decreased

**Return**  $\tilde{x}$

The use of WM for non-quadratic functional is achieved by quadratization. For  $p > 2$ , define  $\widehat{w}_{ij} = w_{ij}(\tilde{x}_i - \tilde{x}_j)^{p-2}$  and the WM follows by substituting  $w_{ij}$  with  $\widehat{w}_{ij}$  in (7) and (10). For the bandwidth problem, where  $p$  should tend to infinity, additional WM sweeps with further increasing of  $p$  are employed at the end of the V-cycle as a postprocessing procedure. More details are provided in Section 4.2.

A more involved example is the workbound reduction problem. Using (4), the respective functional for  $\mathfrak{W}$  can be approximated by

$$wb(\mathfrak{W}, \tilde{x}, \delta) \approx \sum_{i \in \mathfrak{W}} \left( \sum_{\substack{j \in \mathfrak{W} \\ \tilde{x}_j < \tilde{x}_i}} w_{ij}(\tilde{x}_i + \delta_i - \tilde{x}_j - \delta_j)^p + \sum_{\substack{j \notin \mathfrak{W} \\ \tilde{x}_j < \tilde{x}_i}} w_{ij}(\tilde{x}_i + \delta_i - \tilde{x}_j)^p \right)^{2/p} = wb_p(\mathfrak{W}, \tilde{x}, \delta), \quad (11)$$

where  $p$  should tend to infinity so that the longest edges become dominant as desired. The quadratization of (11) is achieved by Taylor expansion up to the third term as follows

$$wb_p(\mathfrak{W}, \tilde{x}, \delta) \approx wb_p(\mathfrak{W}, \tilde{x}, \underline{0}) + \sum_{i \in \mathfrak{W}} \frac{\partial wb_p}{\partial \delta_i}(\mathfrak{W}, \tilde{x}, \underline{0}) \delta_i + \sum_{i, j \in \mathfrak{W}} \frac{\partial^2 wb_p}{\partial \delta_i \partial \delta_j}(\mathfrak{W}, \tilde{x}, \underline{0}) \delta_i \delta_j. \quad (12)$$

Thus, the system of equations to be solved is composed of  $q$  equations of the form  $\frac{\partial wb_p}{\partial \delta_i} = 0$  and constraints (8). In our experiments, this minimization was applied only as a post-processing procedure right after completing the V-cycle for  $\sigma_2(G)$ . Each  $i$ -th iteration of WM was done with sequentially growing even power parameter  $p$ . Since the involved analytic derivatives of (12) are rather lengthy, it is easier and more efficient to use numerical derivatives.

### 3.3.4 Simulated Annealing

A general method to escape false local minima and advance to lower costs is to replace the strict minimization by a process that still accepts each candidate change which lowers the cost, but also assigns a positive probability for accepting a candidate step which increases the cost of the arrangement. The probability assigned to a candidate step is equal to  $\exp(-\Delta/T)$ , where  $\Delta > 0$  measures the *increase* in the arrangement cost and  $T > 0$  is a temperature-like control parameter which is gradually decreased toward zero. This process, known as *Simulated Annealing* (SA) [20], in large problems would usually need to apply *very gradual* cooling (decrease of temperatures), making it extremely slow and inefficient for approaching the global optimum.

In the multilevel framework, however, the role of SA is somewhat different. At each level it is assumed that the *global* arrangement of aggregates has been inherited from the coarser levels, and thus only *local*, small-scale changes are needed. For that purpose, we have started at relatively high  $T$ , lowered it *substantially* at each subsequent sweep, until windows minimization is employed.

Repeated heating and cooling is successively employed for better search over the local landscape. This search is further enhanced by the introduction of a “memory”-like tool consisting of an additional permutation which stores the *Best-So-Far* (BSF) observed arrangement, which is being occasionally updated by a procedure called *Lowest Common*

*Configuration* (LCC) [6]. LCC enables the systematic accumulation of *sub*-permutations over a sequence of different arrangements, such that each BSF sub-permutation exhibits the best (minimal) sub-order encountered so far. The complete description of the SA and LCC algorithms is given in [28].

The entire disaggregation procedure for the minimum  $p$ -sum problem is summarized below in Algorithm 2.

**Algorithm 2:** Disaggregation(coarse level  $\mathcal{C}$ , fine level  $\mathcal{F}$ )

**Parameters:**  $k_2, k_3$

**Decide** on the appropriate power  $p$

**Initialize**  $\mathcal{F}$  from  $\mathcal{C}$

**Apply**  $k_2$  sweeps of compatible relaxation on  $\mathcal{F}$

**Apply**  $k_3$  sweeps of Gauss-Seidel-like relaxation on  $\mathcal{F}$

**Apply** Windows Minimization on  $\mathcal{F}$

**Apply** SA on  $\mathcal{F}$

**If**  $\mathcal{F}$  is the finest level **add** postprocessing of minimization

**Return** the linear order of  $\mathcal{F}$

## 4 Results and Related Works

We have implemented and tested the algorithm using standard C++, LAPACK++ [13] and LEDA libraries [23] on Linux machine. The implementation is non-parallel and not fully optimized.

### 4.1 Previous work

**The Minimum linear arrangement** [28]. We have tested our algorithm on the benchmarks provided by Petit [24] and Koren [21]. Most successful competitive heuristics were : Spectral Sequencing, Optimally Oriented Decomposition Tree, Multilevel based, Simulated Annealing, Genetic Hillclimbing and some of their combinations. The test suite provided in [24] contains rather small graphs for which our algorithm gave the best costs (in almost all cases) in comparison to all previously listed heuristics. The running time was so negligible, that comparison was meaningless. The most interesting result was the comparison of our AMG-like algorithm with the combination of spectral and multilevel approaches [21] on very large graphs (introduced there). The fast version of our algorithm which run only a fifth of the time of [21] exhibited an average improvement of 7%. Our slower but more evolved version improved the costs of [21] by 12%. Other heuristics were not tested on this suite, because of their higher than linear complexity. For complete list of results see [28].

**The Minimum 2-sum** [29]. We have found only one article [16] with an implemented algorithm and numerical results for the minimum 2-sum problem. The algorithm is based on the spectral approach. Since their test suite is relatively small to provide enough information regarding the problem, we have launched a new, much larger test suite and compared our results to the spectral approach. Our multilevel algorithm without any minimization at the

finest level provided much better results (better by an average of 31.4%) than the spectral one. Finally, the minimization process applied after both strategies has proven itself to be good enough for both of the approaches and almost equalized the results. For complete list of results see [29].

## 4.2 Bandwidth

There are many different theoretical and practical results for the bandwidth problem, however, only few allow tests on large inputs. We have thus chosen to test our algorithm on the test suites of [2, 12] which include also large enough inputs to make the picture complete. The authors of [2, 12] present several algorithms that derive a permutation by computing the second eigenvector of the graph’s *Laplacian*  $A$  (a  $|V| \times |V|$  matrix), whose terms are defined by

$$A_{ij} = \begin{cases} -w_{ij} & \text{for } ij \in E, i \neq j \\ 0 & \text{for } ij \notin E, i \neq j \\ \sum_{k \neq i} w_{ik} & \text{for } i = j \end{cases} .$$

The graphs from [2, 12] are presented at the left part of Table 1, along with their best results at column “[2,12]”. Our results (columns “ML5”, “ML10” and “ML150”) are given as ratios to theirs. “ML5” introduces the results obtained by one V-cycle with five WM at all levels (with  $q = 5, 10, 15, 20, 25$ , see Algorithm 1). Note that on the finest level  $p$  is increased by two from one window size to another. We run the algorithm ten times, each starts from a different permutation of the nodes. The best obtained results show an improvement of about 22% over [2, 12]. The means of the ten runs are worse than the corresponding “ML5”-values by an average of 2%, while the standard deviation (around the means) is 1% on the average. We have next tested the outcome of our algorithm with enlarged number of WM. The V-cycle corresponding to “ML10” uses ten WM at all levels (with window sizes 5 to 50 and increased  $p$  only at the finest level) and results with an improvement of 25%, while “ML150” has the same ten WM at each coarse level and 150 iterations at the finest, where  $p$  is increased by two every three iterations of window sizes 5, 10 and 20. (In fact, even though  $p$  in (2) should tend to infinity, in practice, the minimization process has almost not progressed after  $p \approx 100$ .) The “ML150” shows improvement of 32% on the average over [2, 12]. In these two versions, the *means* of the ten runs are worse than the corresponding “ML10”(“ML150”)-values by an average of 2(1.5)%, while the standard deviation (around the means) is 1(0.5)% on the average.

## 4.3 Workbound reduction

Continuing the comparison of multilevel and spectral frameworks started in [29], we present our results for the workbound reduction problem compared to the best known values from [12]. The test suite graphs are the same as in the bandwidth problem. In the second part of Table 1 we present the results we have obtained for these graphs. In column “[12]” we have extracted the *best* results reported in [12]. These results were obtained by several modifications of the spectral sequencing method. Then the results for two types of V-cycles (ten executions for each V-cycle) are presented: the “ $\sigma_2(G)$ ” V-cycle which is aimed at achieving

fast performance and thus somewhat compromising the quality of the arrangement cost by simply approximating the workbound only with the  $\sigma_2(G)$  solution; and the " $\sigma_2(G)$ +WM" V-cycle which starts with the  $\sigma_2(G)$  solution and then applies a postprocessing of 20 additional iterations with increased  $p$  of WM (of sizes 5,10,15,20,25,5,10,...) using (4) and then ten sweeps of node by node minimization using (3). The latter version runs longer but succeeds in finding lower cost arrangements. Our results are presented in the form of ratio between our cost and the best known values from [12]. On the average they exhibit 19% improvement for  $\sigma_2(G)$  and 32% when the postprocessing is added. The *means* of the ten runs are worse than the corresponding " $\sigma_2(G)$ " (" $\sigma_2(G)$ +WM")-values by an average of 2.5(1.5)%, while the standard deviation (around the means) is 1(0.5)% on the average.

Finally, we have also tried to add stochasticity by implementing the SA process. Here as well as for the bandwidth problem we obtained no significant improvement, i.e., no more than the observed variance. Still, as was shown in [28], SA can be extremely important in other problems.

## 4.4 Additional experiments

We have tried to use the minimum 2-sum as a first approximation also for the bandwidth as it was done for the workbound. However, this attempt was unsuccessful. The nature of the bandwidth functional is somewhat different than other  $p$ -sum problems or the workbound. It deals with the minimization of only several concrete edges, those which are the longest, while in the  $p$ -sum and workbound it is necessary to minimize many edges, at least one per node.

As an additional preliminary experiment aimed at checking whether the minimum 2-sum may indeed provide a good first approximation for another functional, we tested it for the *wavefront reduction* problem defined by

$$wf(G, \pi) = \left( \frac{\sum_i |f_i|^2}{n} \right)^{1/2} , \quad (13)$$

where  $f_i = adj(\{\pi^{-1}(1), \dots, \pi^{-1}(i)\}) \cup \{\pi^{-1}(i)\}$  and  $adj(X) = \bigcup_{j \in X} \{k : kj \in E\} \setminus X$ . We have compared our results with those of [18] obtained by a multilevel algorithm. We have just *evaluated* for 15 graphs the wavefront functional on the arrangement produced by the V-cycle with  $p = 2$  and obtained similar results to those presented in [18]. We emphasize that these results are *prior* to any postprocessing which would involve minimization with the particular wavefront functional.

## 5 Conclusions

We have presented a variety of multilevel algorithms for the class of linear ordering problems for general graphs. These algorithms are based on the general principle that during coarsening each vertex may be associated to more than just one aggregate according to some "likelihood" measure. The uncoarsening initialization, which produces the first arrangement of the fine graph nodes, strongly relies on energy considerations (unlike usual interpolation in classical AMG). This initial order is further improved by Gauss-Seidel-like relaxation,

Table 1: Results.

Graph	$ V $	$ E $	[2,12]	ML5	ML10	ML150	[12]	$\sigma_2(G)$	$\sigma_2(G)+WM$
3dtube	45330	1584144	2741	0.76	0.75	0.70	1.54E+11	1.00	0.97
bcsplr08	1624	2213	131	0.66	0.63	0.53	1.23E+06	0.68	0.57
bcsplr09	1723	2394	123	0.69	0.66	0.58	1.41E+06	0.64	0.53
bcsplr10	5300	8271	288	0.68	0.64	0.52	1.43E+07	0.85	0.71
bcsstk12	1423	16342	109	0.63	0.61	0.60	4.59E+06	0.81	0.78
bcsstk13	2003	40940	546	0.72	0.68	0.60	1.63E+08	0.80	0.58
bcsstk24	3562	78174	227	0.88	0.86	0.79	7.42E+07	0.97	0.95
bcsstk29	13830	302424	838	0.68	0.66	0.63	5.26E+09	0.18	0.16
bcsstk30	28924	1007284	2512	0.51	0.49	0.43	4.32E+09	0.91	0.67
bcsstk31	35586	572913	1104	1.06	1.04	0.81	1.97E+10	0.60	0.51
bcsstk32	44609	985046	2339	0.87	0.84	0.66	2.83E+10	0.61	0.47
bcsstk33	8738	291583	519	1.11	1.03	0.97	1.93E+09	0.98	0.87
bcsstk35	30237	709963	1764	0.66	0.64	0.54	1.00E+10	0.74	0.62
bcsstk36	23052	560044	1474	0.68	0.66	0.58	8.52E+09	0.74	0.66
bcsstk37	25503	557737	1373	0.72	0.69	0.59	1.45E+10	0.49	0.44
bcsstk38	8032	173714	669	0.61	0.58	0.54	4.52E+08	0.84	0.69
bcsstm13	649	9949	171	0.66	0.64	0.59	6.50E+06	0.89	0.78
blackhole	2121	6370	105	1.11	1.05	0.97	8.91E+06	0.98	0.86
bus1138	1138	1458	106	0.62	0.60	0.52	7.38E+05	0.64	0.52
bus685	685	1282	83	0.55	0.52	0.47	2.28E+05	0.82	0.71
can_1054	1054	5571	140	0.64	0.64	0.59	2.62E+06	0.99	0.67
can_1072	1072	5686	159	0.82	0.77	0.74	4.08E+06	0.90	0.55
can_445	445	1682	105	0.59	0.58	0.54	9.12E+05	0.95	0.80
can_838	838	4586	126	0.78	0.76	0.70	2.89E+06	0.96	0.64
ct20stif	52329	1273983	3187	0.95	0.91	0.70	1.94E+11	0.38	0.29
dwt_1007	1007	3784	48	0.63	0.60	0.58	4.63E+05	0.99	0.94
dwt_2680	2680	11173	65	1.03	1.00	0.89	3.74E+06	1.00	0.94
dwt_918	918	3233	53	0.68	0.68	0.68	4.68E+05	0.90	0.83
finan512	74752	261120	1331	0.89	0.86	0.83	6.19E+09	0.87	0.64
gearbox	153746	4463329	6271	1.07	0.96	0.89	1.36E+12	0.57	0.42
gupta3	16783	4653322	12535	0.71	0.69	0.66	3.26E+11	1.11	0.99
jagmesh1	936	2664	27	1.22	1.19	1.19	5.38E+05	1.04	1.00
jagmesh9	1349	3876	40	1.00	0.98	0.98	9.82E+05	0.90	0.87
msc10848	10848	609464	1349	0.74	0.73	0.65	3.08E+09	0.96	0.62
msc23052	23052	559817	1524	0.65	0.65	0.57	8.00E+09	0.78	0.69
nasa1824	1824	18692	205	0.85	0.83	0.72	3.13E+07	0.89	0.80
nasa4704	4704	50026	348	0.68	0.65	0.61	1.36E+08	0.98	0.91
pwtk	217918	5653257	2190	0.88	0.80	0.76	2.27E+11	0.67	0.66
shuttle_eddy	10429	46585	177	0.73	0.72	0.68	6.46E+07	0.83	0.74
skirt	12595	91961	309	0.60	0.58	0.50	1.73E+08	0.31	0.26
sstmodel	2730	9702	83	0.96	0.92	0.82	4.72E+06	0.82	0.74
vibrobox	12328	165250	3961	0.57	0.55	0.48	2.70E+10	0.90	0.58
<b>AVERAGE</b>				<b>0.78</b>	<b>0.75</b>	<b>0.68</b>		<b>0.81</b>	<b>0.68</b>

windows minimization and possibly by employing stochasticity, i.e., simulated annealing. The running time of the algorithms is linear, thus it can be applied to very large graphs. In addition, we have proposed two general principles that can be used for different functionals : (1) a first approximation can be obtained from the arrangement produced by one V-cycle of the minimum 2-sum problem instead of using the very popular (but expensive) spectral approach; (2) the continuation approach in which functionals that contain an evaluation of power  $p$  are successively approximated by a sequence of similar but lower power functionals.

Since our algorithms were developed for practical purposes we compared them to many different heuristics : Spectral Sequencing, Optimally Oriented Decomposition Tree, Multi-level based, Simulated Annealing, Genetic Hillclimbing and other. In almost all cases we observed a significant improvement of the results by tens and sometimes by hundreds percents. Our algorithms have proven themselves very stable (i.e., small standard deviations) and of high quality both as a first approximation (using “light” V-cycles) and as more aggressive energy minimizers (with more “heavy” postprocessing).

We recommend our multilevel algorithms as a general practical method for solving linear ordering problems and as a fast and high-quality method for obtaining first approximation for them. The implemented algorithm can be obtained at [30].

## Appendix: Parameters

In order to control the running time of the algorithm it is important to decrease the total number of edges of the constructed coarse graphs. This is achieved by the following two parameters: the maximum allowed coarse neighborhood size  $r$ , which restricts the allowed size  $|N_i|$  of the coarse neighborhood of a vertex  $i \in F$  by deleting the weakest  $w_{ij}$ ,  $j \in C$ ; and the edge filtering threshold  $\epsilon$ , which deletes every *relatively* weak edge  $ij$  (from the created coarse graph) for which both  $w_{ij} < \epsilon \cdot s_i$  and  $w_{ij} < \epsilon \cdot s_j$ , where  $s_i = \sum_k w_{ik}$ .

The specific values of  $r$  and  $\epsilon$  along with those of the three parameters controlling Algorithms 1 and 2 are presented in Table 2. Note that these parameters are the ones used only for the finest levels. As the coarse graphs become much smaller they are accordingly increased. This hardly affects the entire running time of the algorithm but systematically improves the obtained results. In the last column of Table 2 we specifically describe the increase introduced for each parameter as a function of level  $L$ , which usually depends on the ratio  $R = \max(1, |E_0|/|E_L|)$  measuring the relative decrease of the number of edges at level  $L$  compared with the original graph.

We tested many options for the window sizes in Algorithm 1. Usually these sizes were relatively small and very robust to changes. In our implementation we used  $WinSizes = \{5, 10, 15, 20, 25, 30\}$ , however similar results were obtained with other sets of windows, for example,  $WinSizes = \{5, 9, 17, 23, 29\}$ .

## Acknowledgments

This research was supported by

Table 2: The parameters used in the V-cycle.

Parameter	“Value”	The increase for level $L$
The coarse neighborhood size ( $r$ )	10	$+\log(R)$
The edge filtering threshold ( $\epsilon$ )	0.001	$\cdot 0.9^{\log(R)}$
$k_1$ used in the WM	5	$+\log(\sqrt{R})$
The number of sweeps of Compatible relaxation ( $k_2$ )	10	$+2 \cdot L$
The number of sweeps of Gauss-Seidel relaxation ( $k_3$ )	10	$+2 \cdot L$

- Carl F. Gauss Minerva Center for Scientific Computation at the Weizmann Institute of Science;
- The Israel Science Foundation Grant no. 295/01;
- The German-Israel Foundation for scientific research and development (GIF) Grant no. I-718-135.6/2001;
- European Commission Project IST-2002-506766 Aim Shape.

## References

- [1] R. Bar-Yehuda, G. Even, J. Feldman and J. Naor, *Computing an optimal orientation of a balanced decomposition tree for linear arrangement problems*, Journal of Graph Algorithms and Applications, vol. 5, no. 4, pp. 1-27, 2001.
- [2] S.T. Barnard, A. Pothen and H.D. Simon, *A spectral algorithm for envelope reduction of sparse matrices*, Numerical Linear Algebra with Applications, 2(4):317-334, 1995.
- [3] A. Brandt, S. McCormick, and J. Rudge, *Algebraic multigrid (AMG) for automatic multigrid solution with application to geodetic computations.*, Institute for Computational Studies, POB 1852, Fort Collins, Colorado, 1982.
- [4] A. Brandt, S. McCormick, and J. Rudge, *Algebraic multigrid (AMG) for sparse matrix equations.*, In Sparsity and its Applications (Evans, D.J., ed.), Cambridge University Press, Cambridge, 1984, pp. 257-284.
- [5] A. Brandt, *Algebraic Multigrid Theory : The symmetric case*, Appl. Math. Comput., 19:23-56, 1986.
- [6] A. Brandt, D. Ron and D. Amit, *Multi-level approaches to discrete-state and stochastic problems*, Multigrid Methods, II (Hackbush, W. and Trottenberg, U., eds.), Springer-Verlag, 1986, pp. 66-99.
- [7] A. Brandt, *Multiscale Scientific Computation: Review 2001*. In, T. Barth, R. Haimes and T. Chan, eds.: *Multiscale and Multiresolution methods*, Springer-Verlag, 2001. (Proceeding of the Yosemite Educational Symposium, October 2000).

- [8] A. Brandt, *General highly accurate algebraic coarsening*, Gauss Center Report WI/GC-13, May 1999, Electronic Trans. Num. Anal. 10 (2000) 1-20.
- [9] A. Brandt and D. Ron, *Multigrid solvers and multilevel optimization strategies*, in “Multilevel Optimization and VLSICAD” edited by J. Cong and J. R. Shinnerl, Kluwer, 2002.
- [10] W.L. Briggs, V.E. Henson and S.F. McCormick, *A Multigrid Tutorial*, 2nd Edition, SIAM.
- [11] C.K. Cheng, *Linear Placement Algorithms and Applications to VLSI Design*, Networks, vol. 17, pp. 439-464, Winter 1987.
- [12] G. M. Del Corso and F. Romani, *Heuristic spectral techniques for the reduction of bandwidth and work-bound of sparse matrices*, SIAM Journal on Matrix Analysis and Applications, Vol. 18 (4):913–937, 1997.
- [13] J. Dongarra, R. Pozo, and D. Walker, *LAPACK++: A design overview of object-oriented extensions for high performance linear algebra.*, In Proc. Supercomputing '93, pages 162-171. IEEE Computer Soc. Press, 1993.
- [14] J. Diaz, J. Petit, and M. Serna. *A survey on graph layout problems*. ACM Computing Surveys, Volume 34, Issue 3, 2002.
- [15] M.R. Garey, D.S. Johnson, and L. Stockmeyer, *Some Simplified NP-complete graph problems*, Theoretical Computer Science, 1:237-267, 1976.
- [16] A. George and A. Pothen, *An analysis of spectral envelope-reduction via quadratic assignment problems*, SIAM Journal of Matrix Analysis and its Applications, 18(3), pp. 706–732, 1997.
- [17] S. Horton, *The optimal linear arrangement problem: algorithms and approximation*, PhD Thesis, Georgia Institute of Technology, 1997.
- [18] Y. F. Hu and J. A. Scott, *A Multilevel Algorithm for Wavefront Reduction*, SIAM Journal on Scientific Computing, Volume 23, Number 4, pp. 1352-1375, 2001.
- [19] M. Juvan and B. Mohar, *Optimal linear labelings and eigenvalues of graphs*, Discrete Applied Mathematics 36 (1992) 153-168, 1992.
- [20] S. Kirkpatrick, *Models of disordered systems*, Lecture Notes in Physics 149 (C. Castellani et al., eds.), Springer-Verlag, Berlin.
- [21] Y. Koren and D. Harel, *A Multi-Scale Algorithm for the Linear Arrangement Problem*, Proceedings of 28th Inter. Workshop on Graph-Theoretic Concepts in Computer Science (WG'02), Lecture Notes in Computer Science, Vol. 2573, Springer Verlag, pp. 293–306, 2002.
- [22] Y. Lai and K. Williams, *A survey of solved problems and applications on bandwidth, edgesum, and profile of graphs*, J. Graph Theory 31 (1999), 75–94.

- [23] K. Mehlhorn and S. Naher, *LEDA - A platform for combinatorial and geometric computing*, Cambridge University Press, 1999.
- [24] J. Petit, *Experiments for the MinLA problem*, Report de recerca LSI-01-7-R, Departament de Llenguatges i Sistemes Informtics, Universitat Politcnica de Catalunya, 2001.
- [25] T. Poranen, *A genetic hillclimbing algorithm for the optimal linear arrangement problem*, <http://www.cs.uta.fi/tp/optgen/>, 2002.
- [26] D. Ron, S. Wishko-Stern and A. Brandt, *An Algebraic Multigrid based Algorithm for Bisectioning General Graphs*, Technical Report MCS05-01, Department of Computer Science and Applied Mathematics, the Weizmann Institute of Science, 2005.
- [27] J. Ruge, K. Stüben, *Algebraic Multigrid*, In *Multigrid Methods* (McCormick, S. F., ed.), SIAM, Philadelphia, 1987, pp. 73-130.
- [28] I. Safro, D. Ron and A. Brandt, *Graph Minimum Linear Arrangement by Multilevel Weighted Edge Contractions*, *Journal of Algorithms*, in press.
- [29] I. Safro, D. Ron and A. Brandt, *Multilevel Algorithm for the Minimum 2-sum Problem*, submitted, 2004.
- [30] I. Safro, D. Ron and A. Brandt, *Homepage of our projects*, <http://www.wisdom.weizmann.ac.il/~safro>.
- [31] Farhad Shahrokhi, Ondrej Sykora, Laszlo A. Szekly, Imrich Vrto, *On Bipartite Drawings and the Linear Arrangement Problem*, Workshop on Algorithms and Data Structures, 1997.
- [32] E. Sharon, A. Brandt, R. Basri, *Fast Multiscale Image Segmentation*, Proceedings IEEE Conference on Computer Vision and Pattern Proceedings IEEE Conference on Computer Vision and Pattern Recognition, I:70-77, South Carolina, 2000.
- [33] K. Stüben, *An introduction to algebraic multigrid*, Appendix in: *Multigrid* (Trottenberg, U., Oosterlee, C.W. and Schüller, A., eds.), Academic Press, 2001, pp. 413-532.
- [34] K. Stüben, *A review of algebraic multigrid*, *J. Comput. Appl. Math.* 128 (2001) 281-309.