Multilevel Algorithms for Linear Ordering Problems

Ilya Safro
The Weizmann Institute of Science
and
Dorit Ron
The Weizmann Institute of Science
and
Achi Brandt
The Weizmann Institute of Science

Linear ordering problems are combinatorial optimization problems which deal with the minimization of different functionals by finding a suitable permutation of the graph vertices. 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 result for four such problems turned out to be better than every known result in almost all cases, while the short (linear) running time of the algorithms enables testing very large graphs.

Categories and Subject Descriptors: G.2.2 [**Multigrid and multilevel methods**]:

General Terms: Algorithms; Design; Experimentation; Performance; Theory

Additional Key Words and Phrases: multilevel algorithm, linear ordering, algebraic multigrid

1. INTRODUCTION

The objective of the class of linear ordering problems is to minimize different functionals by finding a suitable permutation of the graph vertices [Díaz et al. 2002]. 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 [Safro et al. 2006a] or the bandwidth [Lai and Williams 1999], appear in many applications designed 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 [George and Pothen 1997]. Linear ordering problems may also be motivated as a model used in VLSI design [Cheng 1987] and may be used in several biological applications, graph drawing and other fields (see [Díaz et al. 2002; Lai and Williams 1999; Horton 1997; Shahrokhi et al. 2001]). Commonly for general graphs (or matrices) these problems are NP-hard and their decision versions are NP-complete [Garey et al. 1974].

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 [Juvan and Mohar 1992], optimally oriented decomposition tree [Bar-Yehuda et al. 2001], multilevel based [Koren and Harel 2002; Hu and Scott 2001], simulated annealing [Petit 2003] and others. Some of these algorithms have proven themselves superior in solution quality while others in execution time.

One of the most popular and exploitable methods designed to achieve a suitable linear ordering for different problems is the spectral sequencing (SS) [Juvan and Mohar 1992]. This approach consists of ordering the graph vertices according to the sorted coordinates of the second eigenvector of the graph Laplacian. The heuristic argumentation of SS is based on the fact that the *continuous* version of the minimum 2-sum problem

Corresponding author: Ilya Safro, ilya.safro@weizmann.ac.il

(see Section 2) can be solved by this method to the optimum [Juvan and Mohar 1992]. In practice, for the (discrete) minimum 2-sum it was shown in [Safro et al. 2006b] that the direct application of SS (without additional reinforcement postprocessing) on "real world instances" does not achieve good enough results, while the lower bounds based on SS are very far from the best known ordering costs. Rather poor results of the *exact* SS were presented in [Corso and Romani 2001a] for the minimum bandwidth problem. Better results were shown there by using different *approximated* SS, i.e., by calculating the second eigenvector *less* precisely. In fact, they have tested 19 algorithms (17 of which are different versions of SS) and presented the best achieved results among all. In Section 4 we show the significant improvement achieved by our algorithm over all those algorithms, on the average our results were better by 34%.

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) [Brandt et al. 1982; 1984; Brandt 1986; Briggs et al. 2000; Ruge and Stüben 1987; Stüben 2001a; 2001b]. While in previous works we have developed and tested special multilevel algorithms for solving the minimum linear arrangement problem [Safro et al. 2006a] and the minimum 2-sum problem [Safro et al. 2006b], 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 increasing 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.) [Brandt 2001; Brandt and Ron 2003]. 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 [Sharon et al. 2000] and for the linear arrangement problem [Safro et al. 2006a]. 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 (as will be explained below): 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.

One of the important achievements of our work is the general coarsening that turns out to be suitable for all the different functionals we have tested. This fact can be explained by the way the hierarchy of problems is constructed: variables are eliminated within the coarsening phase only and exactly when they show strong dominant connections to the remaining (non-eliminated) variables, this in turn guarantees that the solution of the eliminated variables is naturally obtained once the non-eliminated variables are solved. The various algorithms thus differ only 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. We have developed a simultaneous minimization of several vertices called Window Minimization. In its basic application (for the 2-sum problem [Safro et al. 2006b]) 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) [Kirkpatrick 1981] 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 will not discuss here theoretical complexity issues, such as lower and upper bounds for the solution cost. We are not interested in worst possible scenarios nor in random instances. Our focus is on practical high-performance and low computational cost algorithms that will outperform existing algorithms by providing better results in less running time. For that purpose we used a known benchmark [Davis 1997] from which

we took graphs of various origins and sizes including very large instances. Our multilevel algorithm exhibits linear complexity, i.e., the computational running time is proportional to $|V| + |E|$.

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%, while the running time for graphs with about 10^4 nodes and 10^5 edges is less than one minute. In general, our experimental results show that the AMG framework can be used for linear ordering problems to obtain high quality results in linear time while using the exact same set of parameters. The implemented algorithm can be downloaded from [Safro].

The paper is arranged as follows. The various functionals and their generalizations are described in Section 2. The multilevel algorithm along with additional optimization techniques are presented in Section 3. A comparison of our results with other works is finally summarized in Section 4.

2. DEFINITIONS AND GENERALIZATIONS

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

$$\pi :\ V \ \longrightarrow\ \{1, 2, ..., n\} \quad .$$

The purpose of linear ordering problems is to minimize some functional over all possible permutations π. The following functional should be minimized for the minimum p-sum problem[1] (MpSP):

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

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. Given the vector of all volumes, v, the task now is to minimize the cost

$$\sigma_p(G, \pi, v) \stackrel{def}{=} \sigma_p(G, x) = \sum_{ij} w_{ij} |x_i - x_j|^p \quad ,$$

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 for unweighted graph ($w_{ij} = 1,\ \forall ij \in E$) can be formulated in term of $\sigma_p(G, \pi)$:

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

since $\sigma_p(G, \pi)$ for large enough p is practically dominated by the longest edge, i.e., by the bandwidth [Juvan and Mohar 1992]. Besides the minimum bandwidth problem there are two NP-hard well known problems defined by MpSP: (a) the minimum linear arrangement problem (where $p = 1$, see [Díaz et al. 2002]) and (b) the minimum 2-sum problem (where $p = 2$, see [George and Pothen 1997]).

The minimization functional of the workbound reduction problem is defined as

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

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

$$wb(G, \pi, v) \stackrel{def}{=} wb(G, x) = \sum_i \max_{j:x_j<x_i} (x_i - x_j)^2 \approx \sum_i \Big(\sum_{j:x_j<x_i} (x_i - x_j)^p \Big)^{2/p} \quad . \tag{4}$$

[1]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.

3. THE ALGORITHM

In the multilevel framework a hierarchy of decreasing size graphs : $G_0, G_1, ..., G_k$ is constructed. Starting from the given graph, $G_0 = G$, create by recursive *coarsening* the sequence $G_1, ..., G_k$, 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. The coarsening used here is similar to the weighted aggregation we have used in solving the minimum linear arrangement and the minimum 2-sum problems [Safro et al. 2006a; 2006b]. We will briefly repeat its basic components for the completeness of the paper.

Coarsening: Weighted Aggregation. The coarsening is interpreted as a process of *weighted aggregation* of the graph nodes to define the nodes of the next coarser graph. In *weighted* aggregation each node can be divided into $fractions$, and different fractions belong to different aggregates. 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 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 of the connections (or edges) between the coarse nodes is calculated.

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 (i.e., a node in F) 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, employing the following steps. As a measure of how large an aggregate seeded by $i \in F$ might grow, define its *future-volume* ϑ_i by

$$\vartheta_i = v_i + \sum_{j \in V} v_j \cdot \frac{w_{ji}}{\sum\limits_{k \in V} w_{jk}} \quad . \tag{5}$$

Nodes with future-volume larger than η times the average of the ϑ_i's are first transferred to C as most "representative". (In our tests $\eta = 2$). The insertion of additional fine nodes to C depends on a threshold Q (in our tests $Q = 0.4$) as specified by Algorithm 1. That is, a fine node i is added to C if its relative connection to C is not strong enough, i.e., smaller than Q. Also, vertices with larger values of ϑ_i are given higher priority to be chosen to belong to C.

Algorithm 1: CoarseNodes($Parameters : Q,\ \eta$)

$C \leftarrow \emptyset,\ F \leftarrow V$
Calculate ϑ_i for each $i \in F$, and their average $\overline{\vartheta}$
$C \leftarrow$ nodes i with $\vartheta_i > \eta \cdot \overline{\vartheta}$
$F \leftarrow V \setminus C$
Sort F in descending order of ϑ_i
Go through all $i \in F$ in descending order of ϑ_i
  If $\left(\sum\limits_{j \in C} w_{ij} / \sum\limits_{j \in V} w_{ij} \right) \leq Q$ **then** move i from F to C
Return C

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 α_i is determined by the demand that $|N_i|$ does not exceed the allowed coarse neighborhood size r chosen to control complexity. (For typical values of r consider the parameters in [Safro et al. 2006b]). 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

$$P_{iI(j)} = \begin{cases} w_{ij} / \sum_{k \in N_i} w_{ik} & \text{for } i \in F,\ j \in N_i \\ 1 & \text{for } i \in C,\ j = i \\ 0 & \text{otherwise} \end{cases} . \quad (6)$$

$P_{iI(j)}$ thus represents the likelihood of i to belong to the $I(j)$-th aggregate. Following the weighted aggregation scheme used in [Sharon et al. 2000], the edge connecting two coarse aggregates, p and q, is assigned with the weight $w_{pq}^{(coarse)} = \sum_{k \neq l} P_{kp} w_{kl} P_{lq}$. The volume of the i-th coarse aggregate is $\sum_j v_j P_{ji}$. Note that during the process of coarsening the total volume of all vertices is conserved.

Coarsest levels. Solving the appropriate functional at the coarsest level is performed by trying all possible arrangements. Since the amount of work investigated at the coarsest levels is negligible compared with that of the finest levels, many solutions can in fact be kept at each level whose graph is relatively small with respect 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 enlarges the range of the search by either extracting different best solutions or combining them using LCC (see below at the end of this section).

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 4.1). Experiments show that the variety of solutions generated thus is similar to those obtained by a single run with multitude of solutions at the coarsest levels, thus it became less important to also use the later. Still this approach has proven to work well for [Ron et al. 2005].

Disaggregation. 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 (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 4.

Before we turn to the details of these common stages of the disaggregation process, let us describe the 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.

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 $y_i \in \{y \ : \ |\sum_{y_j<y,\ j\in V'} w_{ij} - \sum_{y_j>y,\ j\in V'} w_{ij}| \quad \text{is minimal}\}$, i.e., y_i is within the median segment of the nodes in V' to which node i is connected,

—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 node 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

$$x_i \ = \ \frac{v_i}{2} \ + \ \sum_{y_k<y_i} v_k \quad , \tag{7}$$

thus retaining order while taking volume (length) into account. This solution is now *feasible*, i.e., there is no overlap between any pair of nodes.

Relaxation. The arrangement obtained after the initialization is a first feasible solution for the MpSP which is then improved by employing several sweeps of *relaxation*, first *compatible* then *Gauss-Seidel-like* (see Algorithm 2). These two types of relaxation are very similar to the above initialization: The compatible relaxation, 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 (7).

Algorithm 2: Compatible/Gauss-Seidel-like relaxation(current feasible order x)

Initialize $y_i = x_i$ for all $i \in V$
For all $i \in F$ (in case of the Gauss-Seidel-like relaxation $i \in V$)
  If $p = 1$ then y_i has to minimize $|\sum_{y_j<y_i,\ j\in V} w_{ij} - \sum_{y_j>y_i,\ j\in V} w_{ij}|$
  If $p = 2$ then $y_i = \frac{\sum_{j\in V} y_j w_{ij}}{\sum_{j\in V} w_{ij}}$
  If $p > 2$ then y_i has to minimize $\sum_{j\in V} w_{ij}(y_i - y_j)^p$
End
For all $i \in V$ recalculate the new feasible order $x_i = \frac{v_i}{2} \ + \ \sum_{y_k<y_i} v_k$
Return

Window Minimization. The cost of the arrangement can be further reduced by *strict minimization*, i.e., a sequence of rearrangements 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 [Safro et al. 2006a]). 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 *Window 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 have described the basic WM involving the quadratic form for $p = 2$ in [Safro et al. 2006b]. Here we show some possible generalizations. Given a current approximation $\tilde{x}$ to the arrangement of the graph, denote by δ_i a *small correction* to $\tilde{x_i}$. Let $\mathfrak{W} = \{i_1 = \pi^{-1}(s+1), ..., i_q = \pi^{-1}(s+q)\}$ be a *window*, i.e., q successive vertices in the current arrangement, positioned at $\tilde{x}_{i_1}, ..., \tilde{x}_{i_q}$. The local minimization problem of the power p functional associated with a given window $\mathfrak{W}$ can be formulated as follows :

$$\text{minimize } \sigma_p(\mathfrak{W}, \tilde{x}, \delta) = \frac{1}{2}\sum_{i,j\in\mathfrak{W}} w_{ij}(\tilde{x_i} + \delta_i - \tilde{x_j} - \delta_j)^p + \frac{1}{2}\sum_{\substack{i\in\mathfrak{W}\\ j\notin\mathfrak{W}}} w_{ij}(\tilde{x_i} + \delta_i - \tilde{x_j})^p. \tag{8}$$

Since we seek for small corrections δ_i to the current approximation $\tilde{x}_i$, the p-power term in $\sigma_p(\mathfrak{W}, \tilde{x}, \delta)$ can be *quadratized* by evaluating out the *current* $p-2$ power into the coefficient w_{ij}, leaving only quadratic terms as desired. Define $\widehat{w}_{ij} = w_{ij}(\tilde{x_i} - \tilde{x_j})^{p-2}$, then the WM follows by substituting $\widehat{w}_{ij}$ in (8). The new functional $\widehat{\sigma}_2(\mathfrak{W}, \tilde{x}, \delta)$ involved in the corresponding minimization problem is given by

$$\text{minimize } \widehat{\sigma}_2(\mathfrak{W}, \tilde{x}, \delta) = \frac{1}{2} \sum_{i,j \in \mathfrak{W}} \widehat{w}_{ij}(\tilde{x_i} + \delta_i - \tilde{x_j} - \delta_j)^2 + \frac{1}{2} \sum_{\substack{i \in \mathfrak{W} \\ j \notin \mathfrak{W}}} \widehat{w}_{ij}(\tilde{x_i} + \delta_i - \tilde{x_j})^2. \tag{9}$$

To prevent the possible convergence of many coordinates to one point (as the minimization of the energy would tend to do), 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

$$\sum_{i \in \mathfrak{W}} (\tilde{x_i} + \delta_i)^m v_i \;=\; \sum_{i \in \mathfrak{W}} \tilde{x_i}^m v_i \;, \quad m = 1, 2 \tag{10}$$

where for $m = 2$ we have neglected the (presumably small) quadratic term in δ_i. Note that the sums $\sum_{i \in \mathfrak{W}} \tilde{x_i}^m v_i$ for $m = 1, 2$ are invariant under permutations. Using Lagrange multipliers (λ_1 and λ_2), the final formulation of the WM is :

$$\text{minimize } \widehat{\widehat{\sigma}}_2(\mathfrak{W}, \tilde{x}, \delta, \lambda_1, \lambda_2) = \widehat{\sigma}_2(\mathfrak{W}, \tilde{x}, \delta) + \lambda_1 \sum_{i \in \mathfrak{W}} \delta_i v_i + \lambda_2 \sum_{i \in \mathfrak{W}} \delta_i v_i \tilde{x_i} \quad , \tag{11}$$

under the second and third constraints of (12) below, yielding the following system of equations: $\frac{\partial \widehat{\widehat{\sigma}}_2}{\partial x_i} = 0$, for $i = 1, ..., q$ and $\frac{\partial \widehat{\widehat{\sigma}}_2}{\partial \lambda_i} = 0$, for $j = 1, 2$, i.e.,

$$\begin{cases} \sum_{j \in \mathfrak{W}} w_{ij}(\delta_i - \delta_j) + \delta_i \sum_{j \notin \mathfrak{W}} w_{ij} + \lambda_1 v_i + \lambda_2 v_i \tilde{x_i} = \sum_j w_{ij}(\tilde{x_j} - \tilde{x_i}) \quad \text{for } i = 1, ..., q \\ \sum_i \delta_i v_i = 0 \\ \sum_i \delta_i v_i \tilde{x_i} = 0 \quad . \end{cases} \tag{12}$$

Usually in a correct multilevel framework, the changes δ_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 (12), 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 (7). Since the size and location of $\mathfrak{W}$ are quite 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 (7). 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 4.1, 4.2). 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 3 (actual values for q and k_1 are given at the end of the section).

Algorithm 3: WindowMinimization(graph G, current order $\tilde{x}$, window length q, power p)
Parameter: k_1
For $i = 1$ **To** $|V| - q + 1$ **Step** $i = i + \lfloor \frac{q}{2} \rfloor$
$\mathfrak{W} = \{\pi^{-1}(i), ..., \pi^{-1}(i + q - 1)\}$
Store $\tilde{x}|_{\mathfrak{W}}$
Define $\widehat{\widehat{\sigma}}_2(\mathfrak{W}, \tilde{x}, \delta, \lambda_1, \lambda_2)$
Solve the system of equations (12)
Apply k_1 sweeps of Gauss-Seidel-like relaxation on the enlarged $\mathfrak{W}$ with $\tilde{x} + \delta$
Make the solution feasible by applying (7)
Restore $\tilde{x}|_{\mathfrak{W}}$ if the arrangement cost has been increased

Return $\tilde{x}$

For the bandwidth problem, where p should tend to infinity, additional WM sweeps with further increase of p are employed at the end of the V-cycle as a postprocessing procedure. More details are provided in Section 4.1.

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

$$wb(\mathfrak{W},\tilde{x},\delta) \approx \sum_{i\in\mathfrak{W}} \Big(\sum_{\substack{j\in\mathfrak{W}\\ \tilde{x}_j<\tilde{x}_i}} (\tilde{x_i}+\delta_i-\tilde{x_j}-\delta_j)^p + \sum_{\substack{j\notin\mathfrak{W}\\ \tilde{x}_j<\tilde{x}_i}} (\tilde{x_i}+\delta_i-\tilde{x_j})^p \Big)^{2/p} = wb_p(\mathfrak{W},\tilde{x},\delta), \tag{13}$$

where p should tend to infinity so that the longest edges become dominant as desired. The quadratization of (13) is achieved by Taylor expansion up to the third term as follows

$$wb_p(\mathfrak{W},\tilde{x},\delta) \approx wb_p(\mathfrak{W},\tilde{x},\underline{0}) + \sum_{i\in\mathfrak{W}} \frac{\partial wb_p}{\partial\delta_i}(\mathfrak{W},\tilde{x},\underline{0})\delta_i + \sum_{i,j\in\mathfrak{W}} \frac{\partial^2 wb_p}{\partial\delta_i\partial\delta_j}(\mathfrak{W},\tilde{x},\underline{0})\delta_i\delta_j. \tag{14}$$

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 (10). In our experiments, this minimization was applied only as a postprocessing 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 (14) are rather lengthy, it is easier and more efficient to use *numerical* derivatives.

Adding stochasticity. 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 towards zero. This process, known as *Simulated Annealing* (SA) [Kirkpatrick 1981], in large problems would usually need to apply *very gradual* cooling (decrease of temperatures), making it extremely slow and inefficient for obtaining global optimum.

In the multilevel framework, however, the role of SA is somewhat different. At each level it is assumed that the *global* approximate solution has been inherited from the coarser levels, and thus only *local*, small-scale improvements are needed. We have developed a multilevel simulated annealing tool for the linear ordering problems in which the search space for some improvement at every step is small enough to be very local. For that purpose, we have started at relatively high T, lowered it *substantially* at each subsequent sweep until strict minimization is employed.

Repeated heating and cooling is successively employed for better search over the local landscape. This search can be further enhanced by the introduction of a "memory" like tool consisting of an additional permutation which stores the *Best-So-Far* (BSF) observed arrangement. Henceafter, the BSF is being occasionally updated by the procedure called *Lowest Common Configuration* (LCC) [Brandt et al. 1986] which enables the systematic accumulation of *sub*-permutations into it over a sequence of different arrangements, such that each BSF sub-permutation exhibits the best minimal sub-order encountered so far. The cost of the obtained BSF is at most the lowest cost of all the arrangements it has observed, and usually it is lower. The use of LCC becomes more important for larger graphs, where it is expected that the optimum of a subgraph is only weakly dependent on other subgraphs. The BSF is improved by the LCC procedure which updates parts of it taken from the new arrangements reached right after each heating-cooling procedure. All these accumulated updates are thus stored at the BSF which actually provides the current calculated minimum. The complete description of the LCC algorithm is given in [Ron 1990; Safro et al. 2006a; Ron et al. 2005].

The entire disaggregation procedure for the minimum p-sum problem is summarized below in Algorithm 4.

Algorithm 4: 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 Window Minimization on $\mathcal{F}$
Possibly **Apply** Simulated Annealing on $\mathcal{F}$
If $\mathcal{F}$ is the finest level **add** postprocessing of minimization
Return the linear order of $\mathcal{F}$

We tested many options for the window sizes in Algorithm 3. 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, $WinSizes = \{5, 9, 17, 23, 29\}$. The results presented in Tables I and II were received with the following parameters: $k_1 = 5$ (used in the WM); $k_2 = 10$ (the number of sweeps of Compatible relaxation); $k_3 = 10$ (the number of sweeps of Gauss-Seidel relaxation) and with no use of SA.

4. RESULTS AND RELATED WORK

We have implemented and tested the algorithm using standard C++, LAPACK++ [Pozo et al. 1993] and LEDA libraries [Mehlhorn and Näher 1995] on Linux 1.7GHz machine. The implementation is non-parallel and not fully optimized.

Previous work: the minimum linear arrangement and the minimum 2-sum problems [Safro et al. 2006a; 2006b]. We have evaluated our algorithm on the benchmarks provided in [Petit 2003; Koren and Harel 2002; George and Pothen 1997]. Most successful competitive heuristics were : Spectral Sequencing, Optimally Oriented Decomposition Tree, Multilevel based, Simulated Annealing, Genetic Hillclimbing and some of their combinations. Almost all best known results and/or corresponding running time were significantly improved.

4.1 Bandwidth

There are many different theoretical and practical results for the bandwidth problem, e.g., [Campos et al. 2001; Pinana et al. 2004; Caprara and González 2005; Dueck and Jeffs 1995; Martí et al. 2001; Lim et al. 2006], to mention just a few. However, only a small number allow tests on large inputs within a reasonable execution time, e.g., [Barnard et al. 1995; Corso and Romani 2001b; 2001a]. Since we believe that a fair comparison of two heuristics should include final results as well as running times, and since our algorithm is able to deal with very large instances, we have thus chosen to test it on the test suites of [Barnard et al. 1995; Corso and Romani 2001b; 2001a] which include large enough graphs to make the picture complete. These graphs are presented at the leftmost three columns of Table I.

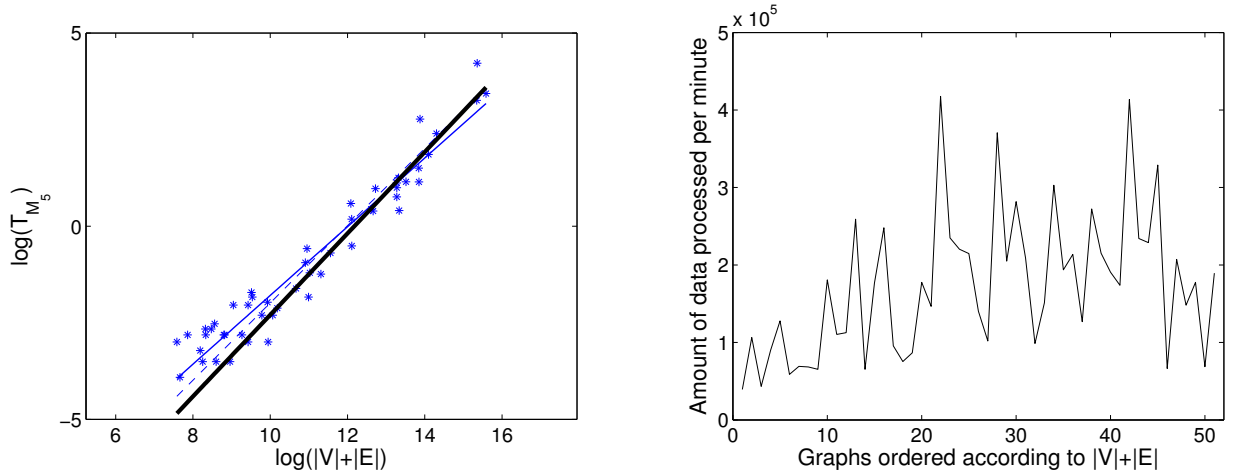
We compare our results to the best results achieved in [Barnard et al. 1995; Corso and Romani 2001b; 2001a], presented at column "bk_∞" of Table I. These results are the best obtained by testing *many* (e.g., 19 in [Corso and Romani 2001a]) *different* algorithms, most of which are versions of the spectral approach. That is, ordering the graph vertices according to the sorted coordinates of the second eigenvector of the graph's *Laplacian* A (a $|V| \times |V|$ matrix), whose terms are defined by

$$A_{ij} \;=\; \begin{cases} -w_{ij} & \text{for } ij \in E,\ i \neq j \\ 0 & \text{for } ij \notin E,\ i \neq j \\ \sum_{k \neq i} w_{ik} & \text{for } i = j \quad . \end{cases} \tag{15}$$

Our results (columns "M_5", "M_{10}" and "M_{200}") are given as ratios to theirs, i.e., to column "bk_∞". "M_5" introduces the results obtained by one V-cycle with five WM at all levels (with $q = 5, 10, 15, 20, 25$, see Algorithm 3). Note that on the finest level p is increased by two from one window size to another. We run the algorithm one hundred times, each starts from a different permutation of the nodes. The best obtained results show an improvement of about 23% over "bk_∞". The means of the one hundred runs are worse than the corresponding "M_5"-values by an average of 7%, while the standard deviation (around the means) is 4.7% on the average. We have next tested the outcome of our algorithm with enlarged number of WM. The V-cycle corresponding to "M_{10}" 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 26%, while "M_{200}" has the same ten WM at each coarse level and 200 iterations at the finest, where p is increased by two every four iterations of window sizes 5, 10, 20 and 40. (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 "M_{200}" shows improvement of 34% on the average over "bk_∞". In these two versions, the *means* of the hundred runs are worse than the corresponding "M_{10}"("M_{200}")-values by an average of 6(4)%, while the standard deviation (around the means) is 3.6(2.3)% on the average.

The running time of one M_5 V-cycle (measured in minutes) is presented in column T_{M_5}. The dependence of the running time on the graph size is depicted in Figures 1-a and 1-b, where in both parts the graphs are ordered through the x-axis according to their size $|V| + |E|$. In part (a), the dependence is presented in logarithmic scale. Each point represents the dependence between the log of the graph size and the log of its running time. Two straight lines fit the experimental points best in a least-squares sense. The bold line (with slope 1.05) corresponds to the experimental points of the largest 25 graphs, while the thin straight line (with slope 0.89) fits the whole set of points. The straight line with slope 1 is presented as a dashed line for comparison purpose. Easy to see that the dependence is very close to linear. The graph (b) roughly shows the amount of data ($|V| + |E|$) processed per minute (i.e., the ratio of values from the respective columns in Table I: $\frac{|V|+|E|}{T_{M_5}}$). In the worst cases only $0.5 \cdot 10^5$ data units are being processed per minute, while for *most* graphs this number varies between 10^5 to $4.2 \cdot 10^5$ units. Clearly, the slower graphs are not necessarily the largest. We do not expect a more precise relationship between $|V| + |E|$ and the running time since the implementation is far from being optimized and since it may depend also on the degree of the graph, on its diameter, etc.



(a) (b)

Fig. 1. Dependence of the running time on the graph size.

In spite of the fact that the real power of the multilevel framework can be observed usually on large instances, where the number of levels is big enough, we have checked the quality of our algorithm on a large set of small graphs. For this goal we have chosen the most recent work [Lim et al. 2006] in which the authors collected the best known values obtained by several algorithms (including their heuristics) on the benchmark of small graphs from [Martí et al. 2001; Pinana et al. 2004]. This benchmark consists of 113 graphs each of them has no more than 1000 vertices. The comparison of quality and running time includes the following heuristics: (1) genetic algorithm and (2) node weighting with shifting algorithm implemented under the hillclimbing framework; and recently developed (3) GRASP and (4) GRASP with Path Relinking methods presented in [Martí et al. 2001; Pinana et al. 2004]. Our numerical results are very similar to the previous best known results from [Lim et al. 2006]. In particular,

$$\text{average}_{G \in \text{[Lim et al. 2006]}} \left(\frac{\text{best known bandwidth from [Lim et al. 2006] on } G}{\text{our best bandwidth on } G} \right) \approx 0.993 \quad . \tag{16}$$

The picture regarding the running time is essentially different

$$\text{average}_{G\in\text{[Lim et al. 2006]}} \left(\frac{\text{running time of GA from [Lim et al. 2006] on } G}{\text{our running time on } G}\right) \approx 28, \tag{17}$$

$$\text{average}_{G\in\text{[Lim et al. 2006]}} \left(\frac{\text{running time of Ns}_a \text{ from [Lim et al. 2006] on } G}{\text{our running time on } G}\right) \approx 80, \tag{18}$$

$$\text{average}_{G\in\text{[Lim et al. 2006]}} \left(\frac{\text{running time of Ns}_b \text{ from [Lim et al. 2006] on } G}{\text{our running time on } G}\right) \approx 250, \tag{19}$$

$$\text{average}_{G\in\text{[Lim et al. 2006]}} \left(\frac{\text{running time of GRASPPR}_a \text{ from [Lim et al. 2006] on } G}{\text{our running time on } G}\right) \approx 78, \tag{20}$$

$$\text{average}_{G\in\text{[Lim et al. 2006]}} \left(\frac{\text{running time of GRASPPR}_b \text{ from [Lim et al. 2006] on } G}{\text{our running time on } G}\right) \approx 235. \tag{21}$$

The quality of (16) can be improved to 0.999 by adding WM sweeps to the postprocessing at the price of doubling the average running time. We may conclude that our multilevel algorithm provides the same results (on the average) as many other heuristics do for small graphs, while presenting an impressive speedup in running time.

We have finally tested our algorithm on the five random graphs appearing in the benchmark [Petit 2003]. We compare a *single* run of our V-cycles with the results of the exact spectral method and with those of the Cuthill-McKee permutation [Cuthill and McKee 1969] which was checked also in [Corso and Romani 2001a]. The results are summarized in Table II showing a clear advantage to our multilevel approach even for those obviously unstructured random graphs.

4.2 Workbound reduction

Continuing the comparison of multilevel and spectral frameworks started in [Safro et al. 2006b], we present our results for the workbound reduction problem versus the best known values from [Corso and Romani 2001b; 2001a]. The test suite graphs are the same as in the bandwidth problem. The results we have obtained for these graphs are presented in the second part of Table 1. In column "bk_{wb}" we have extracted the *best* results reported in [Corso and Romani 2001b; 2001a]. 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 given: 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 [Corso and Romani 2001b; 2001a]. On the average they exhibit 18% improvement for $\sigma_2(G)$ and 31% 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 [Safro et al. 2006a; Ron et al. 2005], SA can be extremely important in other problems.

4.3 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

$$wf(G,\pi) = \Big(\frac{\sum_i |f_i|^2}{n}\Big)^{1/2} \quad , \tag{22}$$

Table I. Comparison of three V-cycles with the previous best known results for the bandwidth and the workbound problems.

			Bandwidth					Workbound		
Graph	$\|V\|$	$\|E\|$	bk_∞	M_5	T_{M_5}	M_{10}	M_{200}	bk_{wb}	σ_2	σ_2+WM
3dtube	4.5E+04	1.6E+06	2334	0.89	11.00	0.87	0.81	1.48E+11	1.04	0.99
add20	2.4E+03	5.4E+03	711	0.60	0.03	0.55	0.50	9.78E+07	0.39	0.20
add32	5.0E+03	7.4E+03	669	0.03	0.05	0.03	0.03	1.67E+07	0.02	0.01
barth	6.7E+03	2.0E+04	200	0.76	0.12	0.72	0.64	4.09E+07	0.99	0.88
barth4	6.0E+03	1.7E+04	213	0.60	0.10	0.58	0.55	3.23E+07	0.76	0.71
barth5	1.6E+04	4.6E+04	370	0.65	0.30	0.63	0.57	1.89E+08	0.93	0.88
bcspwr08	1.6E+03	2.2E+03	131	0.63	0.03	0.63	0.53	1.10E+06	0.76	0.64
bcspwr09	1.7E+03	2.4E+03	123	0.68	0.07	0.65	0.57	1.18E+06	0.76	0.63
bcspwr10	5.3E+03	8.3E+03	288	0.68	0.18	0.63	0.52	1.43E+07	0.85	0.71
bcsstk12	1.4E+03	1.6E+04	109	0.61	0.10	0.61	0.57	4.29E+06	0.87	0.83
bcsstk13	2.0E+03	4.1E+04	546	0.69	0.20	0.64	0.60	1.63E+08	0.80	0.58
bcsstk24	3.6E+03	7.8E+04	227	0.79	0.29	0.80	0.79	7.10E+07	1.01	1.00
bcsstk29	1.4E+04	3.0E+05	838	0.68	1.48	0.67	0.63	1.09E+09	0.85	0.78
bcsstk30	2.9E+04	1.0E+06	2512	0.50	3.15	0.48	0.43	4.32E+09	0.91	0.67
bcsstk31	3.6E+04	5.7E+05	1104	1.14	3.50	1.03	0.78	1.97E+10	0.60	0.51
bcsstk32	4.5E+04	9.9E+05	2339	0.97	4.50	0.87	0.71	2.83E+10	0.61	0.47
bcsstk33	8.7E+03	2.9E+05	519	1.12	1.55	1.03	0.99	1.93E+09	0.98	0.87
bcsstk35	3.0E+04	7.1E+05	1764	0.69	3.16	0.66	0.55	1.00E+10	0.74	0.62
bcsstk36	2.3E+04	5.6E+05	1474	0.70	2.71	0.67	0.57	8.52E+09	0.74	0.66
bcsstk37	2.6E+04	5.6E+05	1373	0.75	3.06	0.70	0.59	1.45E+10	0.49	0.44
bcsstk38	8.0E+03	1.7E+05	669	0.64	0.60	0.58	0.55	4.52E+08	0.84	0.69
bcsstm13	6.5E+02	9.9E+03	171	0.62	0.06	0.62	0.60	6.50E+06	0.89	0.78
blckhole	2.1E+03	6.4E+03	105	1.15	0.13	1.11	0.96	8.91E+06	0.98	0.85
bus1138	1.1E+03	1.5E+03	106	0.61	0.06	0.59	0.51	5.52E+05	0.85	0.69
bus685	6.9E+02	1.3E+03	83	0.47	0.05	0.46	0.42	2.28E+05	0.82	0.70
can1054	1.1E+03	5.6E+03	121	0.74	0.06	0.73	0.67	2.59E+06	1.00	0.67
can1072	1.1E+03	5.7E+03	159	0.81	0.06	0.78	0.74	4.08E+06	0.90	0.55
can445	4.5E+02	1.7E+03	78	0.76	0.02	0.74	0.71	9.12E+05	0.93	0.80
can838	8.4E+02	4.6E+03	126	0.77	0.03	0.75	0.71	2.80E+06	0.98	0.66
ct20stif	5.2E+04	1.3E+06	3187	1.30	6.40	1.26	0.80	1.94E+11	0.38	0.29
dwt1007	1.0E+03	3.8E+03	38	0.76	0.07	0.76	0.74	4.63E+05	0.98	0.94
dwt2680	2.7E+03	1.1E+04	65	0.97	0.16	0.95	0.86	3.74E+06	1.00	0.94
dwt918	9.2E+02	3.2E+03	50	0.72	0.06	0.70	0.68	4.55E+05	0.92	0.85
ex27	9.7E+02	2.0E+04	128	0.96	0.05	0.96	0.95	5.81E+06	1.01	0.77
finan512	7.5E+04	2.6E+05	1331	0.91	2.65	0.87	0.84	6.19E+09	0.87	0.64
gearbox	1.5E+05	4.5E+06	6271	0.68	26.00	0.86	0.65	1.36E+12	0.57	0.42
gupta3	1.7E+04	4.7E+06	12535	0.70	68.00	0.70	0.66	3.26E+11	1.11	0.99
jagmesh1	9.4E+02	2.7E+03	27	1.19	0.04	1.19	1.11	5.38E+05	1.04	1.00
jagmesh9	1.3E+03	3.9E+03	40	0.98	0.08	0.98	0.98	9.82E+05	0.90	0.87
memplus	1.8E+04	4.2E+04	5747	0.85	0.16	0.81	0.59	7.48E+10	0.57	0.15
msc10848	1.1E+04	6.1E+05	1349	0.78	1.50	0.73	0.64	3.08E+09	0.96	0.62
msc23052	2.3E+04	5.6E+05	1524	0.70	2.14	0.64	0.56	8.00E+09	0.78	0.69
nasa1824	1.8E+03	1.9E+04	205	0.80	0.14	0.77	0.73	2.68E+07	1.03	0.93
nasa4704	4.7E+03	5.0E+04	348	0.67	0.39	0.64	0.60	1.36E+08	0.96	0.91
pwt	3.7E+04	1.4E+05	339	0.92	1.20	0.88	0.76	7.51E+08	0.93	0.89
pwtk	2.2E+05	5.7E+06	2190	0.89	31.00	0.86	0.77	2.27E+11	0.67	0.66
shuttleeddy	1.0E+04	4.7E+04	177	0.72	0.56	0.70	0.67	6.46E+07	0.83	0.74
skirt1	1.3E+04	9.2E+04	309	0.60	0.50	0.57	0.50	1.73E+08	0.31	0.26
sstmodel	2.7E+03	9.7E+03	83	0.92	0.13	0.90	0.81	4.72E+06	0.82	0.74
twotone	1.2E+05	9.4E+05	19538	0.77	16.00	0.74	0.67	4.43E+12	0.76	0.65
vibrobox	1.2E+04	1.7E+05	3961	0.60	1.80	0.56	0.46	2.70E+10	0.90	0.58
AVERAGE				**0.77**		**0.74**	**0.66**		**0.82**	**0.69**

where $f_i = adj(\{\pi^{-1}(1), ..., \pi^{-1}(i)\}) \bigcup \{\pi^{-1}(i)\}$ and $adj(X) = \bigcup_{j \in X} \{k : kj \in E\} \backslash X$. We have compared our results with those of [Hu and Scott 2001] 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 [Hu and Scott 2001]. We emphasize that these results are *prior* to any postprocessing which would involve minimization with the particular wavefront functional.

Table II. Results for random graphs.

Graph	$\|V\|$	$\|E\|$	Spectral	Cuthill-McKee	M_5	M_{10}	M_{200}
randomA1	1.0E+03	5.0E+03	828	0.80	0.65	0.59	0.55
randomA2	1.0E+03	2.5E+04	969	0.92	0.91	0.88	0.84
randomA3	1.0E+03	5.0E+04	985	0.95	0.95	0.94	0.90
randomA4	1.0E+03	8.2E+03	855	0.89	0.83	0.75	0.69
randomG4	1.0E+03	8.2E+03	143	0.71	0.54	0.51	0.50

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, window 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) the continuation approach in which functionals that contain an evaluation of power p are successively approximated by a sequence of similar but with lower power functionals; (2) 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 spectral approach.

Since our algorithms were developed for practical purposes we compared them to many different heuristics such as: Spectral Sequencing, Optimally Oriented Decomposition Tree, Multilevel based, Simulated Annealing, Genetic Hillclimbing and other. For almost all instances we observed significant improvement either of the results or of the computational time compared various state-of-the-art methods. Our algorithms have proven themselves to be 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 [Safro].

Acknowledgments

This research was supported by 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

Bar-Yehuda, R., Even, G., Feldman, J., and Naor, J. 2001. Computing an optimal orientation of a balanced decomposition tree for linear arrangement problems. *Journal of Graph Algorithms and Applications 5,* 4, 1–27.

Barnard, S., Pothen, A., and Simon, H. 1995. A spectral algorithm for envelope reduction of sparse matrices. *Numerical Linear Algebra with Applications 2,* 4, 317–334.

Brandt, A. 1986. Algebraic multigrid theory: The symmetric case. *Jornal of Appl. Math. Comp. 19*, 23–56. Preliminary proceedings of the International Multigrid Conference, April 6–8, 1983, Copper Mountain, CO.

Brandt, A. 2001. Multiscale scientific computation: Review 2001. In *Multiscale and Multiresolution methods (Proceeding of the Yosemite Educational Symposium, October 2000)*, T. Barth, R. Haimes, and T. Chan, Eds. Springer-Verlag.

Brandt, A., McCormick, S., and Ruge, J. 1982. Algebraic multigrid (AMG) for automatic multigrid solution with application to geodetic computations. Tech. rep., Institute for Computational Studies, Fort Collins, CO, POB 1852.

Brandt, A., McCormick, S., and Ruge, J. 1984. Algebraic multigrid (AMG) for sparse matrix equations. In *Sparsity and its Applications*, D. J. Evans, Ed. 257–284.

Brandt, A. and Ron, D. 2003. Chapter 1 : Multigrid solvers and multilevel optimization strategies. In *Multilevel Optimization and VLSICAD*, J. Cong and J. R. Shinnerl, Eds. Kluwer.

Brandt, A., Ron, D., and Amit, D. 1986. Multi-level approaches to discrete-state and stochastic problems. In *Multigrid Methods II*, W. Hackbush and U. Trottenberg, Eds. Springer-Verlag, 66–99.

Briggs, W. L., Henson, V. E., and McCormick, S. F. 2000. *A multigrid tutorial: second edition*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Campos, V., Glover, F., Laguna, M., and Martí, R. 2001. An experimental evaluation of a scatter search for the linear ordering problem. *Journal of Global Optimization 21,* 4, 397–414.

Caprara, A. and González, J. J. S. 2005. Laying out sparse graphs with provably minimum bandwidth. *INFORMS Journal on Computing 17,* 3, 356–373.

Cheng, C.-K. 1987. Linear placement algorithm and applications to VLSI design. *Netw. 17,* 4, 439–464.

Corso, G. M. D. and Romani, F. 2001a. Heuristic spectral techniques for the reduction of bandwidth and work-bound of sparse matrices. *Numerical Algorithms 28,* 1–4 (Dec.), 117–136.

Corso, G. M. D. and Romani, F. 2001b. Heuristic spectral techniques for the reduction of bandwidth and work-bound of sparse matrices. Tech. Rep. TR-01-02, Universitá di Piza, Dipartimento di Informatica.

Cuthill, E. and McKee, J. 1969. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th national conference*. ACM Press, New York, NY, USA, 157–172.

Davis, T. 1997. University of florida sparse matrix collection. *NA Digest 97,* 23.

Díaz, J., Petit, J., and Serna, M. 2002. A survey of graph layout problems. *ACM Comput. Surv. 34,* 3, 313–356.

Dueck, G. W. and Jeffs, J. 1995. A heuristic bandwidth reduction algorithm. *Journal of Combinatorial Mathematics and Combinatorial Computing 18*, 97–108.

Garey, M. R., Johnson, D. S., and Stockmeyer, L. 1974. Some simplified np-complete problems. In *STOC '74: Proceedings of the sixth annual ACM symposium on Theory of computing*. ACM Press, New York, NY, USA, 47–63.

George, A. and Pothen, A. 1997. An analysis of spectral envelope reduction via quadratic assignment problems. *SIAM Journal on Matrix Analysis and Applications 18,* 3, 706–732.

Horton, S. B. 1997. The optimal linear arrangement problem: Algorithms and approximation. Ph.D. thesis, Georgia Institute of Technology.

Hu, Y. F. and Scott, J. A. 2001. A multilevel algorithm for wavefront reduction. *SIAM J. Sci. Comput. 23,* 4, 1352–1375.

Juvan, M. and Mohar, B. 1992. Optimal linear labelings and eigenvalues of graphs. *Discrete Appl. Math. 36,* 2, 153–168.

Kirkpatrick, S. 1981. Models of disordered systems. In *Lecture Notes in Physics 149*, C. Castellani, Ed. Springer-Verlag.

Koren, Y. and Harel, D. 2002. Multi-scale algorithm for the linear arrangement problem. *Proceedings of 28th Inter. Workshop on Graph-Theoretic Concepts*.

Lai, Y. and Williams, K. 1999. A survey of solved problems and applications on bandwidth, edgesum, and profile of graphs. *J. Graph Theory 31*, 75–94.

Lim, A., Rodrigues, B., and Xiao, F. 2006. Heuristics for matrix bandwidth reduction. *European Journal of Operational Research 174,* 1, 69–91.

Martí, R., M., L., F., G., and Campos, V. 2001. Reducing the bandwidth of a sparse matrix with tabu search. *Eur. J. Oper. Res. 135,* 2, 211–220.

Mehlhorn, K. and Näher, S. 1995. Leda: a platform for combinatorial and geometric computing. *Commun. ACM 38,* 1, 96–102.

Petit, J. 2003. Experiments on the minimum linear arrangement problem. *ACM Journal of Experimental Algorithmics, 8*.

Pinana, E., Plana, I., Campos, V., and Martí, R. 2004. GRASP and path relinking for the matrix bandwidth minimization. *Eur. J. Oper. Res. 153,* 1, 200–210.

Pozo, R., Dongarra, J. J., and Walker, D. W. 1993. Lapack++: a design overview of object-oriented extensions for high performance linear algebra. In *Supercomputing '93: Proceedings of the 1993 ACM/IEEE conference on Supercomputing*. ACM Press, New York, NY, USA, 162–171.

Ron, D. 1990. Ph.d. thesis. development of fast numerical solvers for problems in optimization and statistical mechanics. Ph.D. thesis, The Weizmann Institute of Science.

Ron, D., Wishko-Stern, S., and Brandt, A. 2005. An algebraic multigrid based algorithm for bisectioning general graphs. Tech. Rep. MCS05-01, Department of Computer Science and Applied Mathematics, The Weizmann Institute of Science.

Ruge, J. and Stüben, K. 1987. *Algebraic Multigrid*. SIAM, 73–130.

Safro, I. Homepage of our projects. *http://www.wisdom.weizmann.ac.il/~safro*.

Safro, I., Ron, D., and Brandt, A. 2006a. Graph minimum linear arrangement by multilevel weighted edge contractions. *Journal of Algorithms 60,* 1, 24–41.

Safro, I., Ron, D., and Brandt, A. 2006b. Multilevel algorithm for the minimum 2-sum problem. *Journal of Graph Algorithms and Applications 10,* 2, 237–258.

Shahrokhi, F., Sýkora, O., Székely, L. A., and Vrto, I. 2001. On bipartite drawings and the linear arrangement problem. *SIAM Journal on Computing 30,* 6, 1773–1789.

Sharon, E., Brandt, A., and Basri, R. 2000. Fast multiscale image segmentation. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*. 70–77.

Stüben, K. 2001a. *An introduction to algebraic multigrid*. Academic Press, 413–532.

Stüben, K. 2001b. A review of algebraic multigrid. *J. Comput. Appl. Math. 128,* 1-2, 281–309.