A Multilevel Algorithm for the Minimum 2-sum Problem

October 5, 2004

Ilya Safro[1] *Dorit Ron* *Achi Brandt*

Faculty of Mathematics and Computer Science
The Weizmann Institute of Science
POB 26, Rehovot 76100, ISRAEL
http://www.wisdom.weizmann.ac.il
ilya.safro@weizmann.ac.il dorit.ron@weizmann.ac.il achi.brandt@weizmann.ac.il

Abstract

The Minimum 2-sum problem is widely used and studied in many practical and theoretical applications. In this paper we present a linear-time algorithm for the problem inspired by the Algebraic Multigrid approach which is based on weighted edge contraction. Our results turned out to be better than every known result in all cases, while the short running time of the algorithm enabled experiments with very large graphs.

[1]Correspondence author. Tel: 972-8-934-4312, Fax: 972-8-934-2945

1 Introduction

The Minimum 2-sum problem (M2sP) belongs to a large family of graph layout problems such as : Bandwidth, Cutwidth, Vertex Separation, Profile of Graph, Sum Cut, etc. The M2sP appears in several applications for solving problems in the large sparse matrix computation, such as finding the minimum linear arrangement [26, 18] or the bandwidth [27]. The M2sP is also 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 [14]. In addition, the M2sP may be motivated as a model used in VLSI design, where at the placement phase it is chosen to minimize the total squared wire length [10]. Commonly for general graphs (or matrices) these problems are NP-hard and their decision versions are NP-complete [13].

The M2sP becomes a simple quadratic optimization problem with a known solution, due to Hall [15], if the restriction on the solution coordinates is relaxed, e.g., the coordinates need not be all integers, as in the case where all vertices are considered to have equal unity volume (see Section 2). Hall has shown in [15] that the eigenvector v_2 which corresponds to the second smallest eigenvalue of the Laplacian of the graph (provided the graph is connected), is the best nontrivial solution to this unrestricted form of the M2sP (subject to some normalization of the solution). Arrangement of the graph vertices according to v_2 is a well known heuristic, usually called the spectral approach, used for many ordering problems like the minimum linear arrangement [18], partitioning [16, 22, 23, 29], envelope reduction of sparse matrices [1, 19], etc.

The spectral approach is also the most studied method for the M2sP problem. George and Pothen [14] have studied the M2sP as they used it for establishing results for the envelope reduction of matrices. They were trying to demonstrate how close are their lower bounds calculation compared with those of the spectral approach. While for some finite element graphs they indeed get close results, for general graphs the gap was profound. They suggested that this gap can be reduced by applying some local reordering (postprocessing) to the obtained results of the spectral approach.

In this paper we present a new multilevel algorithm for the Minimum 2-sum problem based on the Algebraic MultiGrid scheme (AMG) [2, 3, 4, 9, 25, 30, 31]. 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.) [6, 8]. 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 [28] and for the linear arrangement problem [26]. In the context of graphs it is the Laplacian matrix that represents the related set of equations. The main difference between our approach to 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. The aggregation process we use here is similar to the one used for solving the minimum linear arrangement problem [26].

The disaggregation 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 various local reordering methods. In this article we would like to introduce an algorithm for the strict minimization, called Windows Minimization, which is based on the *simultaneous* reordering of several vertices. Then our postprocessing is intensified by Simulated Annealing (SA) [17] which is a general method to escape local minima. In the multilevel framework SA is aimed at searching only for *local* changes that guarantees the preservation of large-scale solution features inherited from coarser levels.

We compared the results obtained by our multilevel algorithm with those of the spectral approach before and after postprocessing. Before the postprocessing the multilevel results are much better than the spectral ones, while after the postprocessing the multilevel is only about 3% better on the average. However, while the complexity of the multilevel algorithm is linear in the size of the graph, the direct eigenvalue calculation is $O(|V|^3)$ and since high accuracy of the results is crucial for obtaining the correct arrangement, it is not likely that an approximation would be sufficient. Our experimental results show that the Algebraic Multilevel framework can be used for the M2sP to obtain high quality results in linear time.

The problem definition and its generalization are described in Sec. 2. The multilevel algorithm along with additional optimization techniques are presented in Sec. 3. A comparison of our results with the spectral approach is finally summarized in Sec. 4.

2 Problem definition and generalization

Given a weighted graph $G = (V,\ E)$, where $V = \{1, 2, ..., n\}$, denote by w_{ij} the non-negative weight of the edge ij between nodes i and j (if $ij \notin E$ then $w_{ij} = 0$). The purpose of the Minimum 2-sum problem is to find a permutation π of the graph nodes such that the cost $\sigma_2(G, \pi) = \sum_{ij} w_{ij}(\pi(i) - \pi(j))^2$ is minimal. 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 $\sigma_2(G, x) = \sum_{ij} w_{ij}(x_i - x_j)^2$, 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.

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 non-representative. Our focus is on practical high-performance algorithm, 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 : $G_0, G_1, ..., G_k$ is constructed, see Figure 1. Starting from the given graph, $G_0 = G$, create by *coarsening* the sequence $G_1, ..., G_k$, then solve the coarsest level directly, and finally uncoarsen the solution back to G. This entire process is called a *V-cycle*.

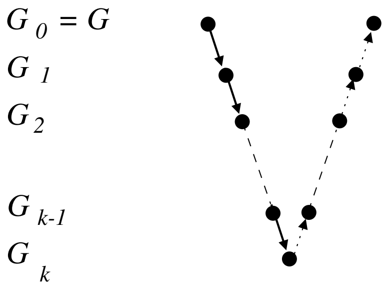

Figure 1: The Scheme of a V-cycle. Solid arrows stand for coarsening, dotted for uncoarsening.

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 problem [26], where coarsening was 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) only 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 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 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.

Let $w_S(ij)$ denote the *normalized* weight of an edge ij with respect to the set of nodes S and to the vertex i, defined by

$$w_S(ij) \;=\; \frac{w_{ij}}{\sum\limits_{k\in S} w_{ik}} \; . \tag{1}$$

As a measure of how large an aggregate seeded by $i \in F$ might grow, define its *future-volume* ϑ_i by

$$\vartheta_i = v_i + \sum_{j\in V} v_j \cdot w_V(ji) \qquad . \tag{2}$$

Nodes with future-volume larger than η times the average of ϑ are first transfered 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 are given higher priority to be chosen to C.

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

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

For convenience we are currently using a library $O(n \cdot log(n))$ sorting algorithm. However, since no exact ordering is really needed, this can be replaced by a rough bucketing sort which has $O(n)$ complexity. This remark will be valid below for all cases where we have used exact sort.

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 $N_i = \{j \in C,\ w_{ij} \ge \alpha_i\}$, 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 Appendix). The classical AMG interpolation matrix P (of size $|V| \times |C|$) is defined by

$$P_{ij} = \begin{cases} w_{N_i}(ij) & \text{for } i \in F,\ j \in N_i \\ 1 & \text{for } i \in C,\ j = i \\ 0 & \text{otherwise} \end{cases} . \tag{3}$$

P_{ij} thus represents the likelihood of i to belong to the j-th aggregate. Let $I(k)$ 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 k. Following the weighted aggregation scheme used in [28], the edge connecting two coarse aggregates, $p = I(i)$ and $q = I(j)$, is assigned with the weight $w_{pq} = \sum_{k \neq l} P_{ki} w_{kl} P_{lj}$. 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.

Solving 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.

3.2 Disaggregation (uncoarsening)

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 quadratic arrangement cost. This approximation is subsequently improved by several *relaxation* sweeps, first compatible, then regular with or without additional stochastic elements, as explained below and summarized in Algorithm 3.

3.2.1 Initialization

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, the coordinate y_i is located at its minimum (volumes are not taken into account)

$$y_i = \frac{\sum_{j \in V'} y_j w_{ij}}{\sum_{j \in V'} w_{ij}}. \tag{4}$$

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

$$x_i = \frac{v_i}{2} + \sum_{y_k < y_i} v_k \quad , \tag{5}$$

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

3.2.2 Relaxation

The arrangement obtained after the initialization is a first feasible solution for M2sP 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 [7], improves the positions of the F-nodes according to the minimization criterion (4) (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 (5).

3.2.3 Windows Minimization

The cost of the arrangement can be further reduced by *strict minimization*, 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 indeed a local operation, 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 (see [26]). This method may also be used for M2sP. However, we would like to propose a more advanced method of local minimization, called *Windows Minimization* (WM), which is suitable for both the multilevel and the spectral approach frameworks. The difference between WM and simple node by node minimization is that WM *simultaneously* minimizes the arrangement cost of several nodes. Given a current approximation $\tilde{x}$ to the arrangement of the graph, denote by δ_i a *correction* to $\tilde{x}_i$. Let $\mathfrak{W} = \{i_1 = \pi^{-1}(p+1), ..., i_q = \pi^{-1}(p+q)\}$ be a *window* of q sequential vertices in the current arrangement, i.e., the nodes positioned at q subsequent coordinates $\tilde{x}_{i_1}, ..., \tilde{x}_{i_q}$. The local energy minimization problem associated with a given window $\mathfrak{W}$ can be formulated as follows :

$$\text{minimize } \sigma_2(\mathfrak{W}, \tilde{x}, \delta) = \sum_{i,j\in\mathfrak{W}} w_{ij}(\tilde{x}_i + \delta_i - \tilde{x}_j - \delta_j)^2 + \sum_{\substack{i\in\mathfrak{W}\\ j\notin\mathfrak{W}}} w_{ij}(\tilde{x}_i + \delta_i - \tilde{x}_j)^2. \qquad (6)$$

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

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

For simplicity, we have used only the first two moments, where for $m = 2$ we have neglected the quadratic term in δ_i. Using Lagrange multipliers, the final formulation of the window minimization problem is :

$$\text{minimize } \sigma_2(\mathfrak{W}, \tilde{x}, \delta, \lambda_1, \lambda_2) = \sigma_2(\mathfrak{W}, \tilde{x}, \delta) + \lambda_1 \sum_{i\in\mathfrak{W}} \delta_i v_i + \lambda_2 \sum_{i\in\mathfrak{W}} \delta_i v_i \tilde{x}_i \quad , \qquad (7)$$

under the second and third constraints of (8) below, yielding the following system of equations:

$$\begin{cases} \sum_{j\in\mathfrak{W}} w_{ij}(\delta_i-\delta_j)+\delta_i\sum_{j\notin\mathfrak{W}} w_{ij}+\lambda_1 v_i+\lambda_2 v_i\tilde{x}_i=\sum_j w_{ij}(\delta_i-\delta_j) \quad \text{for } i=1,...,q \\ \sum_i \delta_i v_i=0 \\ \sum_i \delta_i v_i\tilde{x}_i=0 \quad . \end{cases} \tag{8}$$

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 (8), 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 (5). Since the size and location of $\mathfrak{W}$ are quiet arbitrary, the energy cost of the new sub-arrangement is further improved by Gauss-Seidel-like relaxation sweeps applied to an *enlarged* $\mathfrak{W}$, 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 (5). 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 (in our tests there never was a need for more than 6). Our experiments show that the algorithm is robust to changes in the chosen q's; for complete details consider *WinSizes* in the Appendix. Note, however, that q should be small enough to still guarantee linear execution time of the entire algorithm. The WM is summarized in Algorithm 2.

Algorithm 2: WindowsMinimization(graph G, current order $\tilde{x}$)

```
Parameters: $WinSizes$, $k_2$ (for chosen values, consider the Appendix)

For each $q \in WinSizes$
  For $i = 1$ To $|V|$ Step $i = i + \lfloor\frac{q}{2}\rfloor$
      $\mathfrak{W} = \{\pi^{-1}(i), ..., \pi^{-1}(i+q-1)\}$
      Solve the system of equations (8)
      Apply $k_2$ 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 $\tilde{x}$
```

3.2.4 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) [17], 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.

In particular, $2k+1$ candidate locations are examined for each vertex, each corresponds to moving it some distance l, $0 < |l| \leq k$. The initial temperature $T = T(|l|) > 0$ is calculated apriori for each distance l by aiming at the acceptance of a certain percent of changes (for instance 60%). In detail, the probability of moving a vertex l positions ($l = \pm 1, ..., \pm k$) is $Pr(l) = z \cdot min(1, exp(-\Delta(l)/T(|l|))$, where z is a normalization factor calculated by the demands $\sum_{l=-k}^{k} Pr(l) = 1$ and $Pr(0) = z \cdot min_{l=\pm 1,...,\pm k}(1 - Pr(l)/z)$. In each additional sweep $T(|l|)$ is reduced by a factor, such as 0.6. Typically only three such cooling steps are used.

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) [5]. 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 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. Due to the LCC procedure, it is not necessary to wait in the stochastic annealing process until all minimal sub-permutations are *simultaneously* obtained, which may take exponential time; instead it is sufficient to obtain each such minimal sub-order just once, since henceforth it is guaranteed to appear in the BSF. In detail, the BSF (of a certain level) is initialized by the arrangement obtained at the end of the strict minimization. Then the BSF is improved by the LCC procedure which updates parts of it taken from the new arrangements reached right after each heating-cooling cycle. All these accumulated updates are thus stored at the BSF, which thus represents the current calculated minimum. The complete description of the LCC algorithm is given in [26].

The entire disaggregation procedure is summarized below in Algorithm 3. The Algorithm is divided into two parts: the first approximation and the postprocessing corresponding to the results supported later.

```
Algorithm 3: Disaggregation(coarse level $\mathcal{C}$, fine level $\mathcal{F}$)

  Parameters: $k_1, ..., k_5, \gamma$ (for chosen values consider the Appendix.)

  FIRST APPROXIMATION :
    Initialize $\mathcal{F}$ from $\mathcal{C}$
    Apply $k_1$ sweeps of compatible relaxation on $\mathcal{F}$
```

```
POSTPROCESSING :
  Apply $k_2$ sweeps of Gauss-Seidel-like relaxation on $\mathcal{F}$
  Apply Windows Minimization on $\mathcal{F}$
  Initialize $BSF \leftarrow$ current arrangement of $\mathcal{F}$
  Do $k_3$ cycles of heating and cooling
      Calculate $T(|l|)$ for $l = 1, ..., k_4$
      Do $k_5$ steps
          Apply SA within distance $k_4$ on $\mathcal{F}$
          Decrease all $T(|l|)$ by a factor $\gamma$
      Apply Windows Minimization on $\mathcal{F}$
      Update $BSF \leftarrow LCC(BSF,$ current arrangement of $\mathcal{F})$
Return $BSF$
```

4 Results and Related Works

We have implemented and tested the algorithm using standard C++, LAPACK++ [12] and LEDA libraries [20] on Linux machine. The implementation is non-parallel and not fully optimized.

We have found only one article [14] with an implemented algorithm and numerical results for M2sP. The algorithm is based on the spectral approach. Since this test suite is relatively small to provide enough information regarding M2sP, we have launched a new, much larger test suite which consists of graphs from different areas [11, 21]. These graphs are divided into two groups according to their size : the smaller ones are introduced in Table 1, while the larger ones in Table 2. For all the graphs in Table 1 we compare our results with those of the spectral approach. The numbers in columns 4-5 and 7-11 are in percentage above the cost energy presented at the column **"Quick"** (e.g., the 0.8 appearing for the first graph gd96c in column **"ML"** means that the initial cost energy is $3455 \cdot 1.008$). The first approximation obtained by the multilevel V-cycle, i.e., the arrangement obtained right after applying the compatible relaxation at the finest level is introduced in the column **"ML"** of Table 1. We run the algorithm 100 times (using the parameters specified in the Appendix for the **"Quick"** V-cycle), each starts from a different permutation of the nodes. The best obtained results are presented here. The means of the 100 runs are worse than the corresponding **"Quick"**-values by an average of 0.51%, while the standard deviation (around the means) is 0.69% on the average. The **"ML"** results should be compared to the spectral approach results at column **"SP"** obtained by calculating the second eigenvector of the Laplacian[2] of the graph using MATLAB routine. It is clear that the **"ML"** algorithm provides much better results, better by an average of +31.4% (excluding from the statistics bintree10, in which the improvement

[2]The algebraic representation of a graph is given by its *Laplacian* A (a $|V| \times |V|$ matrix), whose terms are defined by

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

is much larger). Only in one case, the 10-dimensional hypercube, the spectral approach provided a lower cost of -2.7%. However, not only the obtained results are much worse, but also the complexity is much higher: in order to arrange the vertices of the graph, the precision of the second eigenvector coordinates must be at least $O(log|V|)$ and usually much better. This is almost impossible while one uses some approximation algorithm. Our results of the spectral approach were thus obtained with 16-digits precision of an exact algorithm. The experiments with lower precision or with approximation algorithms gave much poorer results. The complexity of an exact calculation of the second smallest eigenvector is $O(|V|^3)$ while the multilevel algorithm is linear in the number of edges.

We have next tested the outcome of our postprocessing on both initial sets of results. Most significant improvement was introduced by applying the Gauss-Seidel-like relaxation, as can be seen in Table 1 column '**"ML+GS"** for the multilevel algorithm and **"+GS"** for the spectral approach. The gap between the two has been reduced, but the spectral approach still provides worse results on the average by 6.1%. Next we have applied the windows minimization which concludes our, so called **"Quick"** V-cycle. Comparing columns **"Quick=ML+GS+WM"** with the corresponding **"+WM"** shows that the multilevel results remain better, the spectral ones are worse by an average of 4.5%.

Finally, we introduced stochastisity by applying Simulated Annealing. In the multilevel framework, the SA enters at all levels of the V-cycle. We refer to this version as the **"Extended"** V-cycle (its complete parameters are given in the Appendix). While the **"Quick"** V-cycle is aimed at achieving fast performance, the **"Extended"** V-cycle runs longer but succeeds in finding lower cost arrangements on the average by 1%. The means of the 100 runs of the **"Extended"** V-cycles are worse than the corresponding **"Quick"**-values by an average of 0.49% and the average of the calculated standard deviations (around the means for 100 runs) of the **"Extended"** V-cycle is 0.66%. In column **"+SA"** of Table 1 we present the results obtained after adding SA to the spectral approach followed by the above postprocessing. The improvement is again of only 1%. Our last test was to run a very long SA after the postprocessing with the spectral approach, aiming at achieving comparable amount of work to 100 **"Extended"** V-cycles. These results are given in column **"Heavy SA"**. While improvement is naturally observed, the results on the average remain worse by about 3%, while for 6 graphs out of 38 it is worse by more than 5%.

To enrich our test suite, we present in Table 2 our **"Quick"** V-cycle results for additional 28 relatively large graphs. No spectral approach results are provided since we were not able to run (on the computers avaliable to us) the MATLAB routine and calculate the needed eigenvector. Each result is again the best observed out of 100 runs, for which the means for 100 runs are worse than the corresponding **"Quick"**-values by an average of 0.55% and the average standard deviation is 0.47%.

Table 1: Results.

Graph	$\lvert V\rvert$	$\lvert E\rvert$	**ML**	**ML+GS**	**Quick=ML+GS+WM**	**SP**	**+GS**	**+WM**	**+SA**	**Heavy SA**
gd96c	65	125	0,8	0,3	3,45500E+03	46,0	5,8	0,0	0,0	0,0
gd95c	62	144	0,8	0,0	3,75500E+03	26,2	0,3	0,0	0,0	0,0
gd96b	111	193	8,7	0,0	1,90860E+04	53,7	1,3	1,1	1,0	1,0
gd96d	180	228	8,2	1,0	5,47390E+04	87,0	1,0	0,2	0,1	0,0
dwt245	245	608	2,9	0,4	6,32810E+04	80,4	2,6	0,8	0,8	0,0
bintree10	1023	1022	11,2	0,0	1,35656E+05	16394,2	38,8	11,3	10,3	6,4
bus685	685	1282	9,6	0,2	2,15744E+05	44,9	9,8	7,0	7,0	6,6
bus1138	1138	1458	6,7	0,4	5,52111E+05	76,5	7,4	1,8	0,9	0,4
gd96a	1096	1676	13,6	0,1	1,49741E+07	124,4	31,3	25,2	21,9	15,9
can445	445	1682	1,2	0,0	1,65431E+06	6,0	0,8	0,6	0,6	0,6
c1y	828	1749	8,2	0,0	7,86685E+06	121,1	5,6	4,6	4,3	4,2
c2y	980	2102	7,3	0,0	1,07286E+07	61,2	0,9	0,6	0,3	0,3
bcspwr08	1624	2213	5,6	0,4	9,39437E+05	48,6	13,0	11,2	9,1	1,9
bcspwr09	1723	2394	5,8	0,5	1,01801E+06	71,8	25,5	22,5	22,0	11,3
c5y	1202	2557	7,2	0,0	1,39958E+07	130,5	10,8	9,6	8,4	6,4
jagmesh1	936	2664	2,5	0,1	8,68459E+05	14,2	12,6	12,2	11,8	1,1
c3y	1327	2844	7,0	0,0	1,97321E+07	142,7	7,5	2,5	0,8	0,7
c4y	1366	2915	8,2	0,0	1,66028E+07	51,8	2,1	1,4	0,2	0,0
dwt918	918	3233	5,1	0,1	8,25233E+05	11,5	1,4	0,9	0,3	0,0
dwt1007	1007	3784	1,4	0,0	1,02750E+06	4,3	2,2	1,9	1,7	0,0
jagmesh9	1349	3876	5,2	0,2	1,39541E+06	10,3	6,3	4,5	1,6	0,9
can838	838	4586	0,4	0,0	7,43012E+06	1,8	0,1	0,1	0,0	0,0
randomA1	1000	4974	34,9	1,8	2,96618E+08	49,7	18,2	9,7	4,9	1,1
hc10	1024	5120	3,6	0,0	1,78957E+08	0,8	0,1	0,1	0,0	0,0
can1054	1054	5571	0,2	0,1	6,36257E+06	2,0	0,1	0,1	0,0	0,0
can1072	1072	5686	3,6	0,0	8,70400E+06	3,6	0,1	0,0	0,0	0,0
randomG4	1000	8173	6,9	0,0	7,70221E+06	7,8	1,1	0,8	0,6	0,1
randomA4	1000	8177	18,3	4,3	6,78008E+08	31,2	15,3	5,7	0,9	0,4
bcspwr10	5300	8271	10,5	0,2	1,37238E+07	19,0	4,9	4,1	3,0	2,3
bcsstm13	649	9949	0,5	0,0	3,94573E+07	31,7	0,8	0,6	0,3	0,0
dwt2680	2680	11173	4,2	0,0	9,18901E+06	5,2	0,3	0,1	0,0	0,0
airfoil1	4253	12289	8,9	0,1	1,63343E+07	18,4	7,2	5,9	2,7	1,1
bcsstk12	1423	16342	7,7	0,1	2,06281E+07	19,2	11,9	10,2	6,8	5,9
nasa1824	1824	18692	5,8	0,0	1,41216E+08	24,0	7,9	4,2	1,1	0,3
randomA2	1000	24738	12,8	4,1	2,95112E+09	12,9	5,1	0,3	0,1	0,0
nasa2146	2146	35052	5,3	0,1	1,23584E+08	6,6	4,3	4,2	4,0	2,1
bcsstk30	28924	1007284	3,1	0,0	5,10942E+10	3,9	1,3	1,2	1,1	0,5
bcsstk13	2003	40940	3,4	0,0	6,71461E+08	40,2	14,8	9,3	3,0	2,7

Table 2: Results (cont.)

Graph	$\|V\|$	$\|E\|$	**Quick**	**Graph**	$\|V\|$	$\|E\|$	**Quick**
whitaker3	9800	28989	6,53876E+07	**msc23052**	23052	559817	6,58277E+10
zcrack	10240	30380	1,36392E+08	**bcsstk36**	23052	560044	6,58053E+10
shuttleeddy	10429	46585	1,36209E+08	**bcsstk31**	35586	572913	7,45410E+10
randomA3	1000	49820	6,63583E+09	**msc10848**	10848	609464	5,92180E+10
nasa4704	4704	50026	7,54457E+08	**ferotor**	99617	662431	2,67776E+11
bcsstk24	3562	78174	9,06950E+08	**bcsstk35**	30237	709963	7,51880E+10
bcsstk38	8032	173714	3,87606E+09	**598a**	110971	741934	3,85388E+11
finan512	74752	261120	1,00967E+10	**bcsstk32**	44609	985046	1,46284E+11
bcsstk33	8738	291583	2,97010E+10	**144**	144649	1074393	1,55347E+12
bcsstk29	13830	302424	1,06444E+10	**ct20stif**	52329	1273983	6,77425E+11
ocean	143437	409593	1,16890E+11	**m14b**	214765	1679018	1,67209E+12
tooth	78136	452591	3,18756E+11	**mrng2**	1017253	2015714	1.93775e+13
mrng1	257000	505048	6,69398E+11	**auto**	448695	3314611	1,33598E+13
bcsstk37	25503	557737	6,77934E+10	**pwtk**	217918	5653257	2,25527E+12

5 Conclusions

We have presented a multilevel algorithm for the Minimum 2-sum problem for general graphs. The algorithm is 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, windows minimization and possibly by employing stochasticity, i.e., simulated annealing. The running time of the algorithm is linear, thus it can be applied to very large graphs.

We have compared our results to those obtained by the spectral approach. The calculation of the second eigenvector of the Laplacian of the graph has to be of high accuracy to provide reasonable results. Such a direct computation is of complexity $O(|V|^3)$. Still, the obtained results are much worse than the initial results obtained by our multilevel V-cycle by 31.4% on the average for the smaller sized test suite. In addition, we have applied postprocessing to both initial arrangements. The Gauss-Seidel-like relaxation improves both results most significantly. The windows minimization further reduces the arrangement cost for some graphs. The final results show that the multilevel framework achieves better results of 4.5% on the average. Finally, we have added stochastisity to both algorithms. Both results were improved by about 1%. We have also tried to apply a very long SA to the final results of the postprocessing of the spectral approach. Many results have been further improved, however, some graphs (6 out of 38) still present results higher by more than 5%.

Our main conclusion is that the average and the best results of our V-cycles are better than the results of the spectral approach. We recommend our multilevel algorithm as a general practical method for solving the Minimum 2-sum problem and as a fast and high-quality method for obtaining first approximation for it. The implemented algorithm can be

Table 3: The parameters used for the “quick” and “extended” V-cycles.

Parameter	“quick” V-cycle	“extended” V-cycle	The increase for level L
The coarse neighborhood size (r)	10	10	$+log(R)$
The edge filtering threshold (ϵ)	0.001	0.001	$\cdot 0.9^{log(R)}$
The number of sweeps of Compatible relaxation (k_1)	5	10	$+2 \cdot L$
The number of sweeps of Gauss-Seidel relaxation (k_2)	5	10	$+2 \cdot L$
The number of heating and cooling in SA (k_3)	0	3	$\cdot log(R)$
k_4 used in the SA	0	5	$+log(\sqrt{R})$

obtained at *http://www.wisdom.weizmann.ac.il/~safro/min2sum*.

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 ϵ threshold, which deletes every *relatively* weak edge ij (from the created coarse graph) if both $w_{ij} < \epsilon \cdot s_i$ and $w_{ij} < \epsilon \cdot s_j$, where $s_i = \sum_k w_{ik}$.

These two parameters and five others which control the uncoarsening procedure (see Algorithm 3) are presented in Table 3 for the “quick” and “extended” V-cycles we have used. The last two parameters (of Algorithm 3) were constantly chosen to be $k_5 = 4$ and $\gamma = 0.6$.

It is however important to mention 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 3 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 2. 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, $WinSizes = \{5, 9, 17, 23, 29\}$.

Acknowledgements

This research was supported by a Grant from the German-Israeli Foundation for Scientific Research and Development (G.I.F.), Research Grant Agreement No. I-718-135.6/2001, and by the Carl F. Gauss Minerva Center for Scientific Computation at the Weizmann Institute of Science.

References

[1] 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.

[2] 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.

[3] 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.

[4] A. Brandt, *Algebraic Multigrid Theory : The symmetric case*, Appl. Math. Comput., 19:23-56, 1986.

[5] 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.

[6] 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).

[7] A. Brandt, *General highly accurate algebraic coarsening*, Gauss Center Report WI/GC-13, May 1999, Electronic Trans. Num. Anal. 10 (2000) 1-20.

[8] 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.

[9] W.L. Briggs, V.E. Henson and S.F. McCormick, *A Multigrid Tutorial*, 2nd Edition, SIAM.

[10] C.K. Cheng, *Linear Placement Algorithms and Applications to VLSI Design*, Networks, vol. 17, pp. 439-464, Winter 1987.

[11] T. Davis, *Sparse matrix collection*, http://www.cise.ufl.edu/research/sparse/sparse.

[12] 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.

[13] M.R. Garey, D.S. Johnson, and L. Stockmeyer, *Some Simplified NP-complete graph problems*, Theoretical Computer Science, 1:237-267, 1976.

[14] 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.

[15] K. M. Hall, *An r-dimensional Quadratic Placement Algorithm*, Management Science 17 (1970), 219-229.

[16] B. Hendrickson and R. Leland, *An improved spectral graph partitioning algorithm for mapping parallel computations*, SIAM J. Sci. Comput., 16 (1995), pp. 452-469.

[17] S. Kirkpatrick, *Models of disordered systems*, Lecture Notes in Physics149 (C. Castellani et al., eds.), Springer-Verlag, Berlin.

[18] 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.

[19] G. Kumfert and A. Pothen, *A refined spectral algorithm to reduce the envelope and wavefront of sparse matrices.*, accepted by BIT, 1996.

[20] K. Mehlhorn and S. Naher, *LEDA - A platform for combinatorial and geometric computing*, Cambridge University Press, 1999.

[21] J. Petit, *Approximation heuristics and benchmarkings for the MinLA problem*, In R. Battiti and A. Bertossi, editors, Alex '98, Building bridges between theory and applications, pages 112-128. Universit di Trento, 1998.

[22] A. Pothen, H.D. Simon and K.P. Liou, *Partitioning sparse matrices with eigenvectors of graphs*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 430-452.

[23] A. Pothen, H.D. Simon and L. Wang, *Spectral nested dissection*, Tech. Rep. CS-92-01, Computer Science, Pennsylvania State Univ., University Park, PA, 1992.

[24] D. Ron, *Ph.D. Thesis. Development of fast numerical solvers for problems in optimization and statistical mechanics*, The Weizmann Institute of Science, 1989.

[25] J. Ruge, K. Stüben, *Algebraic Multigrid*, In Multigrid Methods (McCormick, S. F., ed.), SIAM, Philadelfia, 1987, pp. 73-130.

[26] I. Safro , D. Ron and A. Brandt, *Graph Minimum Linear Arrangement by Multilevel Weighted Edge Contractions*, submitted, 2003.

[27] I. Safro , D. Ron and A. Brandt, *Multilevel Algorithm for the Minimum Bandwidth Problem*, to appear.

[28] 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.

[29] D.A. Spielman and S-H. Teng, *Spectral partitioning works: Planar graphs and finite element meshes*, Manuscript, 1996. Avaliable on Web at the URL http://cs.berkley.edu/ spielman/spect.html.

[30] 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.

[31] K. Stüben, *A review of algebraic multigrid*, J. Comput. Appl. Math. 128 (2001) 281-309.