ARTICLE IN PRESS JDA:344

Journal of Discrete Algorithms ••• (••••) •••–•••

Contents lists available at ScienceDirect

Journal of Discrete Algorithms

www.elsevier.com/locate/jda

Multiscale approach for the network compression-friendly ordering

Ilya Safro [*], Boris Temkin

Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, USA

A R T I C L E   I N F O

Article history:
Received 29 April 2010
Received in revised form 17 August 2010
Accepted 29 September 2010
Available online xxxx

Keywords:
Multiscale algorithms
Minimum logarithmic arrangement
Network compression

A B S T R A C T

We present a fast multiscale approach for the network minimum logarithmic arrangement problem. This type of arrangement plays an important role in the network compression and fast node/link access operations. The algorithm is of linear complexity and exhibits good scalability, which makes it practical and attractive for use in large-scale instances. Its effectiveness is demonstrated on a large set of real-life networks. These networks with corresponding best-known minimization results are suggested as an open benchmark for the research community to evaluate new methods for this problem.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

Finding a suitable compressed representation of large-scale networks has been intensively studied in both practical and theoretical branches of data mining [12,2,11,5,26]. In particular, the success of applying some of the recently proposed compression schemes [11,5,3] strongly depends on the "compression-friendly" arrangement of network nodes. Usually, the goal of these arrangements is to order the nodes such that the endpoints of network links (edges) are located as close as possible. Doing so leads to a more compact representation of links and allows a better performance of compression schemes and network element access operations.

In [11], Chierichetti et al. propose a combinatorial optimization problem, namely, *the minimum logarithmic arrangement problem* (MLogA), that minimizes the gap encodings of edges stretched between their endpoints. This is achieved by ordering the network nodes and assigning to them unique integer values (ids) such that the endpoints of a link will obtain close values. Several other well-known optimization problems (such as the minimum linear arrangement and the minimum bandwidth [14]) have goals similar to MLogA. In contrast, however, MLogA seeks a nearly optimal information-theoretical compressed encoding size for all network links as it minimizes the total number of bits to spend for this purpose. For example, instead of the popular network/matrix compressed row representation, which contains a sorted lists of neighbors per node, various link gap encodings can be used (see Table 1). In practice, it often means that further compression algorithms will benefit if locality and similarity properties of a network will be reflected in the MLogA ordering (for deep explanations and background see [2,11,5]). The MLogA problem was proven to be NP-hard [11], and two heuristics for ordering the social networks were given. In this paper, we present a multiscale method for approximating a generalized *link-weighted* and *node-volumed* version of MLogA for *general* networks. The importance of a link-weighted property (when each link is assigned by a nonnegative weight) for this problem is twofold. First, the link weight can measure the significance of that link. For example, in many cases we have information regarding its access frequency, and we would prefer that frequently accessed links would be compressed better. Second, the multiscale algorithmic framework admits a natural aggregation of *weighted*

[*] Corresponding author.
E-mail addresses: safro@mcs.anl.gov (I. Safro), boris.temkin@gmail.com (B. Temkin).
URL: http://www.mcs.anl.gov/~safro (I. Safro).

1570-8667/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.jda.2010.09.007

Table 1
Example of network/matrix representations.

Row/node number	Compressed row format: *sorted neighbors*	Gap format: *gaps*
1	$i, j, k, \ldots$	$i, j-i, k-j, \ldots$
2	$p, q, r, \ldots$	$p, q-p, r-q, \ldots$

links at different scales of the network structure (discussed later). Similarly to the link weights, all nodes are assigned by nonnegative volumes that represent the length of their segments captured in the ordering of the network nodes.

Our multiscale algorithm is based on the algebraic multigrid (AMG) [9] methodology for linear ordering problems [14,24]. The main objective of the AMG-based framework is to construct a hierarchy of problems (*coarsening*), each approximating the original problem but with fewer degrees of freedom. This is achieved by introducing a sequence of successive projections of networks' graph Laplacians into lower-dimensional spaces and solving the problem in them. The multiscale framework has two key advantages that make it attractive for applying on modern large-scale instances: it exhibits a linear complexity, and it can be relatively easily parallelized and implemented by using standard matrix–vector operations. Another advantage of the multiscale framework is its heterogeneity, expressed in the ability to incorporate external appropriate optimization algorithms (as a refinement) in the framework at different scales. For more detailed surveys of multiscale methods and their parallelization for combinatorial optimization problems, we refer the reader to [8,27,7].

In contrast to [11], the main goal of this work is to provide a *generic* MLogA solver for general networks. Social networks (including big parts of web graphs) commonly consist of a combination of structural properties (such as degree distribution, small diameter, and expander-like topology) that enable fast greedy methods (usually based on some preferential ordering of graph traversal) to find arrangements for further high-quality compression. These methods can be successful because most of the links exhibit good locality [18,28] and only a small number of them can be considered as global edges that connect well-separated nodes/regions of a network. Thus, these networks can hardly be considered as strongly irregular and difficult instances. In real life, however, there occur many situations when the structure of a stored network (or a part of it) is complex and irregular (such as decision/detailed supply/infection spread networks and even parts of social networks that do not exhibit power law degree distribution), which creates many contradictions between greedy local decisions and solutions that consider a more global picture. We address this type of problem by introducing a multiscale solver.

The experimental part of this work shows how far the existing state-of-the-art ordering heuristics are from being optimal. We demonstrate significant improvement for minimization of logarithmic arrangement for various families of networks, including social networks and other (ir)regular instances. The comparison was performed with several methods recently introduced in [1,5,6,11]. In almost all cases, our solver exhibited better numerical results than previous best-known results.

We introduce notation and necessary definitions in Section 2. The main algorithm is described in Section 3. The computational results and discussion about different parameter settings are presented in Section 4. In Section 5 we conclude and provide possible future research directions. For readers interested in their own implementation, we note that some details regarding the multiscale scheme (that are not related strongly to the discussed problem) are omitted in this paper; they can be found in [23,24,20,8].

2. Notation and problem definition

A network is described by a weighted directed graph $G = (V, E)$, where $V = \{1, 2, \ldots, n\}$ is the set of nodes (vertices) and E is the set of directed edges. If $ij \in E$, then there exists an edge $i \to j$. Denote by w_{ij} the nonnegative weight of the directed edge ij between nodes i and j; if $ij \notin E$, then $w_{ij} = 0$. Let π be a bijection

$$\pi : V \to \{1, 2, \ldots, n\}.$$

The purpose of the link-weighted version of MLogA is to minimize

$$\sum_{ij \in E} w_{ij} \lg \bigl|\pi(i) - \pi(j)\bigr| \tag{1}$$

over all possible permutations π. The base of a logarithm (lg) will be always 2. We define the generalized form of this problem (GMLogA) that emerges during the multiscale 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 $c(G, x)$, namely,

$$\begin{aligned} &\min_{\pi} \mathrm{c}(G, x) = \sum_{ij \in E} w_{ij} \lg |x_i - x_j| \\ &\quad \text{such that} \quad x_i = \frac{v_i}{2} + \sum_{k, \pi(k) < \pi(i)} v_k. \end{aligned} \tag{2}$$

The constraint ensures that each vertex is positioned at its center of mass, capturing a segment on the real axis that equals its length. The original form of the problem is the special case where all the volumes are equal. Besides the practical importance (explained in Section 1), the generalized form of MLogA is meaningful in the multiscale framework. The existence of

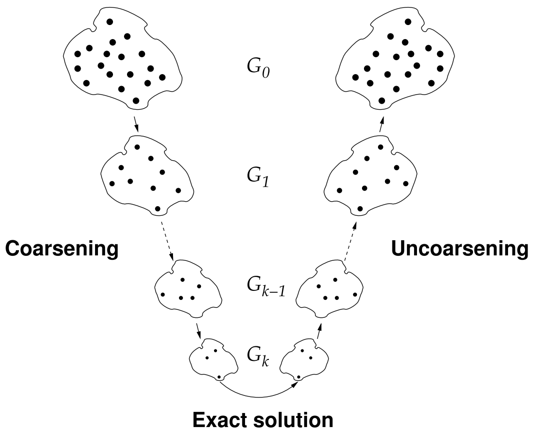


Fig. 1. Scheme of V-cycle. The arrows show the order in which smaller graphs are created and revised for approximation.

node volumes and edge weights allows them to be aggregated: different parts of fine nodes are merged into small clusters (or coarse nodes), and different fine edges are aggregated into the coarse edges that represent the connections between these clusters. In particular, the node volumes and the edge weights quantify the process of the aggregation at different scales.

We denote the process of creating a weighted undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ by $\mathcal{G} \leftarrow \mathrm{un}(G)$, where G is a given directed graph. By applying $\mathrm{un}(G)$, we obtain $\mathcal{V} \leftarrow V$ and

$$ij \in \mathcal{E} \quad \Leftrightarrow \quad ij \in E \text{ or } ji \in E.$$

The edge weights are accumulated from the set of directed edges

$$\forall ij \in \mathcal{E} \quad w_{ij} = \sum_{ij \in E} w_{ij} + \sum_{ji \in E} w_{ji},$$

allowing use of multigraphs. Both MLogA and GMLogA are formulated for the undirected graphs similarly to their directed versions. In this paper we will find approximate solutions for the following problem.

Problem 2.1. Given a directed weighted network graph G, the objective is to find a permutation π that minimizes $\mathrm{c}(G, x_\pi)$, where x_π is a vector of the respective node coordinates restricted by π.

Note that the problem is formulated for both directed and undirected graphs. We define[1] β_{G,x_π} as the main unit for the empirical comparison in Section 4:

$$\beta_{G,x_\pi} = \mathrm{c}(G, x_\pi) / \sum_{ij \in E} w_{ij}. \tag{3}$$

3. The algorithm

In the multiscale framework we construct a hierarchy of decreasing-size graphs, $G_0, G_1, \ldots, G_k$, by *coarsening*, starting from the given graph $G_0 \leftarrow \mathrm{un}(G)$. In other words, we create an undirected version of a given directed graph by assigning to the undirected edges bigger weights if, in the original unweighted graph, the directed edges had been reciprocal. At the coarsest level Problem 2.1 is solved exactly, and starting from the $(k-1)$th level it is formulated and approximated by successive prolongation of the solution from the previous coarser level. This entire process is called a *V-cycle* (see Fig. 1).

3.1. Coarsening

In the present work, coarsening is interpreted as a modified process of weighted aggregation reinforced by the algebraic distance couplings for logarithmic sum minimization. For a detailed discussion about the algebraic distance-based weighted aggregation and multiscale graph organization related to the general graph optimization problems, we refer the reader to [20]. We will briefly repeat its basic components for the completeness of the paper and will concentrate on the modifications related to Problem 2.1.

[1] β with no subindexes will be used where appropriate.

3.1.1. Algebraic distance coupling

Algebraic distance-based coupling is a measure of connectivity strength between two nodes connected by an edge [20, 10]. Given the Laplacian of a graph, denoted by $L = D - W$, where W is a weighted adjacency matrix of a graph and D is the diagonal matrix with entries $d_{ii} = \sum_j w_{ij}$, we define an iteration matrix H for Jacobi overrelaxation (JOR) as

$$H = (D/\omega)^{-1}\big((1/\omega - 1)D + W_l + W_u\big),$$

where $0 \leqslant \omega \leqslant 1$ (see Appendix A) and W_l and W_u are the strict lower and upper triangular parts of W, respectively.

Definition 3.1. The algebraic distance coupling ρ_{ij} is defined as

$$\rho_{ij} = 1\bigg/\left(\sum_{r=1}^{R} \lg\big|\chi_i^{(k,r)} - \chi_j^{(k,r)}\big|\right),$$

where $\chi^{(k,r)} = H^k\chi^{(0,r)}$ is a relaxed randomly initialized test vector (i.e., $\chi^{(0,r)}$ is a random vector sampled over $[-1/2, 1/2]$), R is the number of initial test vectors, and k is the number of iterations.

Note that this definition is a modified version of the original definition from [20,10], making it more suitable for Problem 2.1. Several interesting properties (including convergence and model description) still can be proved similarly to [10].

3.1.2. Coarse graph construction

Considering ρ_{ij} as an edge strength measure, one can construct a coarse graph by defining a classical AMG interpolation matrix P that will project the fine graph to the coarse graph (i.e., to the lower-dimensional space). The projection is represented as

$$L_c \leftarrow P L_f P^T,$$

where L_f and L_c are the Laplacians of fine (G_f) and coarse (G_c) graphs, respectively.

We begin by selecting a set of seed nodes $C \subset V_f$ that will represent the centers of future coarse nodes (or aggregates). In fact, C is interpreted as a dominating set of V_f (not necessarily of minimum size) such that other (fine) nodes $F = V_f \setminus C$ should be strongly coupled to C. This selection can be done by traversing all nodes and leaving those nodes i in F that satisfy

$$\sum_{j\in C}\rho_{ij}\bigg/\sum_{j\in V_f}\rho_{ij} \geqslant \Theta_1 \quad \text{and} \quad \sum_{j\in C} w_{ij}\bigg/\sum_{j\in V_f} w_{ij} \geqslant \Theta_2, \tag{4}$$

where $\Theta_{(1,2)}$ are the parameters of coupling strength (see Appendix A). The order in which the nodes are traversed is based on the *future volume* principle [20], which is a measure of how large (representative in terms of the current minimization problem) a coarse node can be. To keep the linear complexity of the entire framework, the order need not be calculated exactly but only roughly (for example, by using bucket sort). In contrast to the multiscale approach for the generalized minimum p-sum problem [20], we found that automatically moving to C those nodes that have exceptionally large future volume is not necessary and even has a small negative impact in several social networks. The reason rests with the structural properties of many social networks with power-law (or similar) node degree distribution. In many cases, the best position of a node with exceptionally high degree depends on the better location of its neighbors and not vice versa. In other words, if some high-degree node i will be identified as a C-node and its low-degree neighbors will be (partially) aggregated with it, then i will be interpolated back before its neighbors. Thus, the contribution of its low-degree neighbors to the total arrangement cost will be minimized by assigning them close to i. Since other than $\cdot i$ edges can be stretched too much for many low-degree neighbors of i, it can be beneficial for the multiscale frameworks to avoid moving all high-degree nodes to C automatically.

After identifying C, we define for each $i \in F$ its coarse neighborhood N_i^c that contains a limited set of C-nodes to which i is connected. The criterion for choosing C-nodes to N_i^c is also based on ρ_{ij}. 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 is defined by

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

$P_{iI(j)}$ thus represents the likelihood of i belonging to the $I(j)$th aggregate. The edge connecting two coarse aggregates p and q is assigned with the weight $w_{pq} = \sum_{k\neq l} P_{kp} w_{kl} P_{lq}$. The volume of the pth coarse aggregate is $\sum_j v_j P_{jp}$.

```
Input: $x_1, \dots, x_k$ – samples in increasing order
Input: $h$ – smoothing parameter (window)
Input: $p_1, \dots, p_k$ – posteriori probabilities
Output: $\hat{d}_1, \dots, \hat{d}_k$ – estimated density
/* calculate incrementally influence of left neighbors */
$s_1 \leftarrow p_1$
for $t = 2$ to $k$ do
    $s_t = p_t + s_{t-1} 2^{(x_{t-1} - x_t)/h}$
end
/* calculate incrementally influence of right neighbors */
$r_k \leftarrow p_k$
for $t = k - 1$ to $1$ do
    $r_t = p_t + s_{t+1} 2^{(x_t - x_{t+1})/h}$
end
/* aggregate results */
for $t = 1$ to $k$ do
    $\hat{d}_t = s_t + r_t - p_t$
end
```

Algorithm 1: Linear algorithm for density estimation.

3.2. Uncoarsening

The uncoarsening process starts by solving Problem 2.1 at the coarsest level. Since the number of nodes at the coarsest level is very small (in our tests it was 9), the problem can be solved exactly by exhaustive search.

3.2.1. Minimizing the contribution of one node

Before proceeding to the stages of coarse-to-fine projection of a coarse-level solution, we describe how to approximate a solution of Problem 2.1 for a single node only. This will be a basic step in the initialization and relaxation described in Sections 3.2.2 and 3.2.3.

Denote by N_i the set of ith neighbors with already assigned coordinates $\tilde{x}_j$. To minimize the local contribution of i to the total energy (2), we assign to it a coordinate x_i that minimizes

$$\sum_{j \in N_i} w_{ij} \lg |x_i - \tilde{x}_j|. \tag{6}$$

Since for every $j \in N_i$, $x_i = \tilde{x}_j$ implies that the sum (6) is minus infinity, we resolve this ambiguity by setting

$$x_i = \tilde{x}_t \quad \Leftrightarrow \quad t = \arg\min_{k \in N_i} \sum_{k \neq j \in N_i} w_{kj} \lg |\tilde{x}_k - \tilde{x}_j|. \tag{7}$$

The trivial exact solution has a running time $O(|N_i|^2)$, as it requires computing $|N_i|$ sums, each one with $|N_i| - 1$ terms. Thus, to preserve the linear complexity of the entire algorithm, one can use the trivial solution for nodes with small $|N_i|$ only. We will approximate (7) using a heuristic that seeks the nearly minimum sum in the point of maximal density.

Consider set $\{\tilde{x}_j\colon\ j \in N_i\}$ as independent and identically distributed samples of a random variable with unknown distribution and w_{ij} as a posteriori probability of a sample $\tilde{x}_j$. Assuming that, we have to choose a point where the estimated probability density is maximized. Various approaches exist for density estimation [25]. One of the most popular is called the “Parzen window” (or kernel density estimation) method. In this method, the density at point x is estimated as

$$d(x) = \frac{1}{|N_i|h} \sum_{j \in N_i} K\left(\frac{|x - \tilde{x}_j|}{h}\right), \tag{8}$$

where K is a kernel and h is a smoothing parameter called the bandwidth. To simplify calculations, we choose a kernel similar to the Gaussian kernel. Thus, the estimated density at point x is

$$\hat{d}(x) = \frac{1}{|N_i|h} \sum_{j \in N_i} w_{ij} 2^{-|x - \tilde{x}_j|/h}.$$

Note that the factor $\frac{1}{|N_i|h}$ may be omitted without changing the maximal’s location. Moreover, it is sufficient to calculate $\hat{d}(\tilde{x}_j)$ for every $j \in N_i$. This approach allows us to compute the density in all $|N_i|$ points in linear time using Algorithm 1.

Likewise, for all density estimations techniques the crucial step is to choose the bandwidth h. On the one hand, it should be big enough to smooth the peaks and high-oscillatory components. On the other hand, if h is too big, then the maximum density will always be located in the middle. Based on experiments, we choose $h = N/2\lg(N)$, where $N = \max(\tilde{x}_i) - \min(\tilde{x}_i)$.

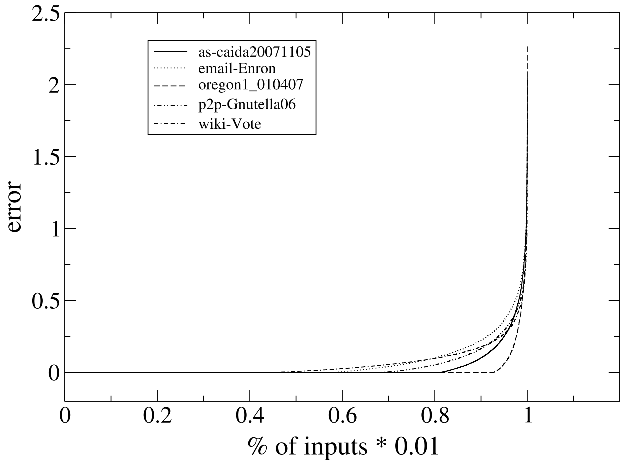


Fig. 2. Error distribution for density approximation on real-life networks. Each curve corresponds to one network. Each point on a curve corresponds to one difference between the exact (quadratic) solution of (7) and its approximation. The comparisons (points) are ordered by their errors in an ascending order.

```
Input: arrangement of a coarse graph
Output: initial ordering of fine level nodes
/* Initialize C-points */
for all $j \in C$ do
    $x_j \leftarrow x_{I(j)}$
end
/* Initialize F-points */
for all $j \in F$ do
    $x_j \leftarrow$ find best coordinate for $j$ using Algorithm 1 with $N_j$
end
/* Legalization of the coordinates */
for all $j \in V$ do
    $x_i = v_i/2 + \sum_{x_k < x_i} v_k$
end
```

Algorithm 2: Initialization of the fine level.

To justify the chosen heuristic, we present our results from experiments on real data. In Fig. 2 we show the error distribution for our approximation. The error is calculated as

$$\text{error} = \hat{\theta} - \theta^*,$$

where θ^* is the energy (excluding $-\infty$ terms) at the exact solution of (7) and $\hat{\theta}$ is an approximation. We can easily see that most of the time the approximation error is either zero or very small.

3.2.2. 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 initialize the fine-level arrangement by letting each seed $j \in C$ inherit the position of its respective aggregate: $x_j \leftarrow x_{I(j)}$. However, in contrast to the *initialization by stages* (that was done for the minimum p-sum and the minimum workbound problems [24]) in which the F-nodes' positions are calculated according to *all* already initialized fine-level neighbors, we found that the initialization that is inspired by the principles of the classical AMG coarse solution projection $x^f \leftarrow P^T x^c$ can produce significantly better approximations. In other words, after initializing the fine-level C-nodes, F-nodes will be placed one by one using Algorithm 1, which takes into account only the C-neighbors of a current F-point. The entire initialization scheme is presented in Algorithm 2.

3.2.3. Relaxations and strict minimization

The initialization produces a feasible (that satisfies the constraint of (2)) solution inherited from the coarse level. This solution is further improved by employing a small number of relaxation sweeps of two types: *compatible* and *Gauss–Seidel* (GS). The goal of the compatible relaxation is to improve the positions of F-nodes while keeping the C-nodes invariant. One sweep of the compatible relaxation can be described by the two last "for" cycles of Algorithm 2 when in the first cycle the best coordinate is calculated over *all* neighbors.

Having the improved ordering of F-nodes relative to the C-nodes, we apply GS relaxation. This relaxation traverses *all* nodes one by one and tries to find a best position for every node such that its own contribution to the total sum of logarithms will be minimized. In all our experiments we observed that, after improving the order with the compatible

```
Input: $L_f$ is the Laplacian of undirected graph $\mathcal{G}$ (initially obtained from un($G$))
Output: approximated graph minimum logarithmic arrangement
if $\mathcal{G}$ is small enough then
    solve the problem exactly
    return the arrangement
end
/* Coarsening                                                    */
calculate algebraic distances $\rho_{ij}$, $\forall ij \in \mathcal{E}$
identify the set of $C$-points (see (4))
construct the fine-to-coarse projection $P$ (see (5))
$L_c \leftarrow P L_f P^T$
/* Recursive call                                                */
$\Pi \leftarrow$ V-cycle($L_c$)
/* Uncoarsening                                                  */
$\pi_1 \leftarrow$ initialization($\Pi$)
$\pi_2 \leftarrow$ compatible-relaxation($\pi_1$)
$\pi_3 \leftarrow$ GS-relaxation($\pi_2$)
$\pi_4 \leftarrow$ N-N refinement($\pi_3$)
return $\pi_4$
```

Algorithm 3: ms-GMLogA: Full scheme of one V-cycle for GMLogA.

relaxation, the GS relaxation has almost no influence on the order. This situation is in contrast to other multiscale linear ordering methods in which the GS relaxation was one of the most powerful and crucial components.

At the end of any iteration of initialization, GS, and compatible relaxation we legalize the order by correcting the coordinates of nodes to achieve the feasible solution of (2). Clearly, the noncompliance of constraint of (2) leads to immediate high concentrations of nodes in only several points. In general, it is not meaningless to leave more than one node at the same coordinate (for example, when they have equal neighborhoods) at the coarse levels and, thus, to ensure for them the same interpolation to the next-finer level. However, it will require a careful identification of what can be the overlapping nodes. In this case, the first problem will be related to preventing fast, high concentrations of nodes in one point. The second major problem will be to update all distances for all endpoints of edges spanning over these overlapping nodes as these edges have to reflect the additional stretching. This updating can require additional data structure for maintaining the edges over different types of cuts.

3.2.4. Strict minimization

We applied two methods of strict minimization (SM, also known as “local refinement”): node-by-node minimization (N-N) and window minimization (WM). The principal difference between relaxations and SM is that in SM each change is accepted if and only if it reduces the entire sum ((2), preserving the feasibility of ordering) and not the local node contribution only (6).

In the N-N minimization all nodes are scanned according to their current order, and each vertex i, in its turn, is checked for the best position over some small enough segment that i belongs to. The k left and k right candidate positions are scanned, and the one with the minimal cost is chosen.

In the WM (see [24]), the total cost of the arrangement is reduced by numerical minimization of a collective contribution of a small group of consecutive nodes. Given a current feasible solution $\tilde{x}$ of the graph logarithmic arrangement, denote by δ_i a small correction to $\tilde{x}_i$. Denote by $\mathfrak{W} = \{i_1 = \pi^{-1}(s+1), \ldots, i_q = \pi^{-1}(s+q)\}$ a subset of successive q vertices in the current arrangement. The goal of the local minimization problem is then to find δ such that

$$\sum_{i,j \in \mathfrak{W}} w_{ij} \lg |\tilde{x}_i + \delta_i - \tilde{x}_j - \delta_j| + \sum_{\substack{i \in \mathfrak{W} \\ j \notin \mathfrak{W}}} w_{ij} \lg |\tilde{x}_i + \delta_i - \tilde{x}_j| \tag{9}$$

is minimized. The solution to this optimization problem can be approximated numerically by using techniques similar to those presented in Section 3.2.1.

The situation with strict minimization is similar to that with GS. In all our experiments, we found no need to use the more complicated WM. The simple N-N minimization with small distance parameter $k < 5$ was enough to obtain the best results. This is a major advantage because in most multiscale schemes the significant part of the running time is spent on the uncoarsening. The lack of need (or, more correctly, optionality) for GS and WM (or other collective optimization) also strongly advocates the choice of the C-nodes because the further compatible relaxation solves the problem well.

Having defined all the components of the V-cycle, we present the full scheme in Algorithm 3.

4. Computational results

The implementation of Algorithm 3 is based on the multiscale framework used in [24]. The implementation is nonparallel and has not been fully optimized. The results (arrangements and running times) should be considered only qualitatively and can certainly be further improved by more advanced implementation and multiscale techniques.

Because of the practical significance of MLogA, we designed an open site [22] with the benchmark graphs and a set of numerical results. In this site we present the results listed below and invite the scientists who work on this problem to submit interesting new networks, their solutions, and improved arrangements for the existing networks.

Our benchmark consists of 100 graphs of different nature and size (most of them are taken from [13] and [17]). For these graphs we created their undirected versions (by applying $\mathtt{un}(\cdot)$) and evaluated our algorithms on both the directed and undirected versions. We evaluated two baseline algorithms:

- MinLA + N-N: the multiscale solver that finds an approximation of MinLA [23] followed by GMLogA-oriented N-N fast postprocessing (see Section 3.2.4 with $k = 10$);
- ms-GMLogA: the GMLogA solver described in Algorithm 3.

Clearly, MinLA and GMLogA can have different optimal orderings. However, we observed that qualitative MinLA orderings reinforced by appropriate postprocessing can also lead to good solutions of GMLogA. In spite of the fact that most of the best solutions were obtained with ms-GMLogA and because the observation was done on a benchmark of large-scale real-life graphs (and there exists intensive research on MinLA [14,16]), we decided to present these results too. We also mention that the Fiedler vector-based solutions (even those that try to achieve a good local minimum for MinLA) usually produce significantly fewer qualitative solutions even with appropriate postprocessing.

In our experiments we use the WebGraph framework [5], which provides a simple way to manage large graphs. In particular, the most recent version of WebGraph produces random, lexicographical [5], Gray [6], double shingle [11], and LayeredLPA [1] orderings for the graphs. The important feature of these orderings is that they are completely endogenous (i.e., determined by the graph itself), contrary to natural ordering, that is, the ordering in which the graph was created. It would seem that the random ordering always has to be significantly worse, than the lexicographic and natural orderings. This is not always true, however, mostly because of the ways the latter have been produced. For example, in highly parallel systems with shared memory, the parallelized graph traversal algorithms can produce arrangements that have been dumped in parallel from different processors. Each processor produces an arrangement of good locality for a small subset of nodes, but the entire arrangement does not possess this property. In particular, in our benchmark we found two graphs whose random orderings were at least comparable to the natural ones.

We use the average number of bits per link β (3) as a unit of comparison for the main results presented in Figs. 3–6. In Figs. 3 and 4 the left and right columns correspond to the comparison results of directed and undirected graphs, respectively. In Figs. 3(a)–(f), each plot area consists of two curves. Each point in bold curves represents a ratio between $\mathtt{c}(G)$ obtained by ms-GMLogA and $\min(\mathtt{c}(G, \pi_{nat}), \mathtt{c}(G, \pi_{lex}), \mathtt{c}(G, \pi_{rnd}))$ for a particular graph from the benchmark, where π_{nat}, π_{lex}, and π_{rnd} are the best orderings obtained by natural, lexicographic, and randomized arrangements, respectively. Similarly, each point in regular curves corresponds to the ratio with MinLA + N-N in the numerator. To demonstrate the robustness of the algorithm, we show several sets of its parameters. Each row pair of subfigures corresponds to the same set of parameters for directed and undirected graphs, respectively.

In the multiscale algorithms, there are several possible sources (see [8]) of potentially higher than linear complexity (or linear with coefficients that are too big). Here we address the four most important: (a) the number of iterations in the relaxations/refinements, (b) the complexity of one iteration of relaxation/refinement, (c) the order of interpolation that can increase the complexity of the coarse graphs, and (d) too many C-nodes (addressed in Appendix A). The first set of parameters (Figs. 3(a), (b)) (which is a suggested default set) contains a mild configuration. The order of interpolation (number of nonzero entries in one row of P) is only 1; k in N-N refinement is 5; and the number of relaxations of any kind is at most 20. The calculation of the algebraic distance couplings in this set is based on 5 randomly initialized vectors and only 20 iterations of JOR. For almost all graphs, we observed that ms-MLogA produces better orderings than does MinLA + N-N. On average, ms-MLogA improves the graphs by more than 40% of their ordering cost in comparison to natural, lexicographic, and random arrangements.

We note that the most beneficial graphs (those that have ratios from 0.2 to 0.4) in both directed and undirected versions come from different collections such as VLSI design networks, part of the web network, road maps, and Amazon links. In general, these graphs differ significantly in their structural properties. We observed that no particular part of the benchmark was less beneficial than another, thus attesting to the generality of the proposed method.

The difference between improvements obtained with different parameter sets is not significant; however, we provide them to demonstrate the robustness of the method. In the second set of parameters (Fig. 3(c), (d)) the number of random initial vectors in the algebraic distance coupling is reduced to 1 only. It leads to the faster coarsening (while keeping all convergence and model properties of the algebraic distance [10]) and potentially weaker decisions regarding edge strengths based on the algebraic distance. In the third set of parameters (Fig. 3(e), (f)) we increase k in N-N refinement from 5 to 10 and the interpolation order from 1 to 2 at the finest level of hierarchy. At all coarse levels these parameters are increased, too, according to the logarithmic increase scale described in [23].

In Fig. 4(a), (b) we show the difference between the very fast version of ms-GMLogA with no refinement and fast relaxation (5 sweeps) and the slowest version with very aggressive relaxation/refinement parameters ($k = 25$ and 40 sweeps). Each subfigure contains two curves: regular and bold. The regular corresponds to the fast version and the bold to the slowest one. It is easy to see that the difference between these two versions is not significant and one can certainly use a fastest version to obtain qualitative results.

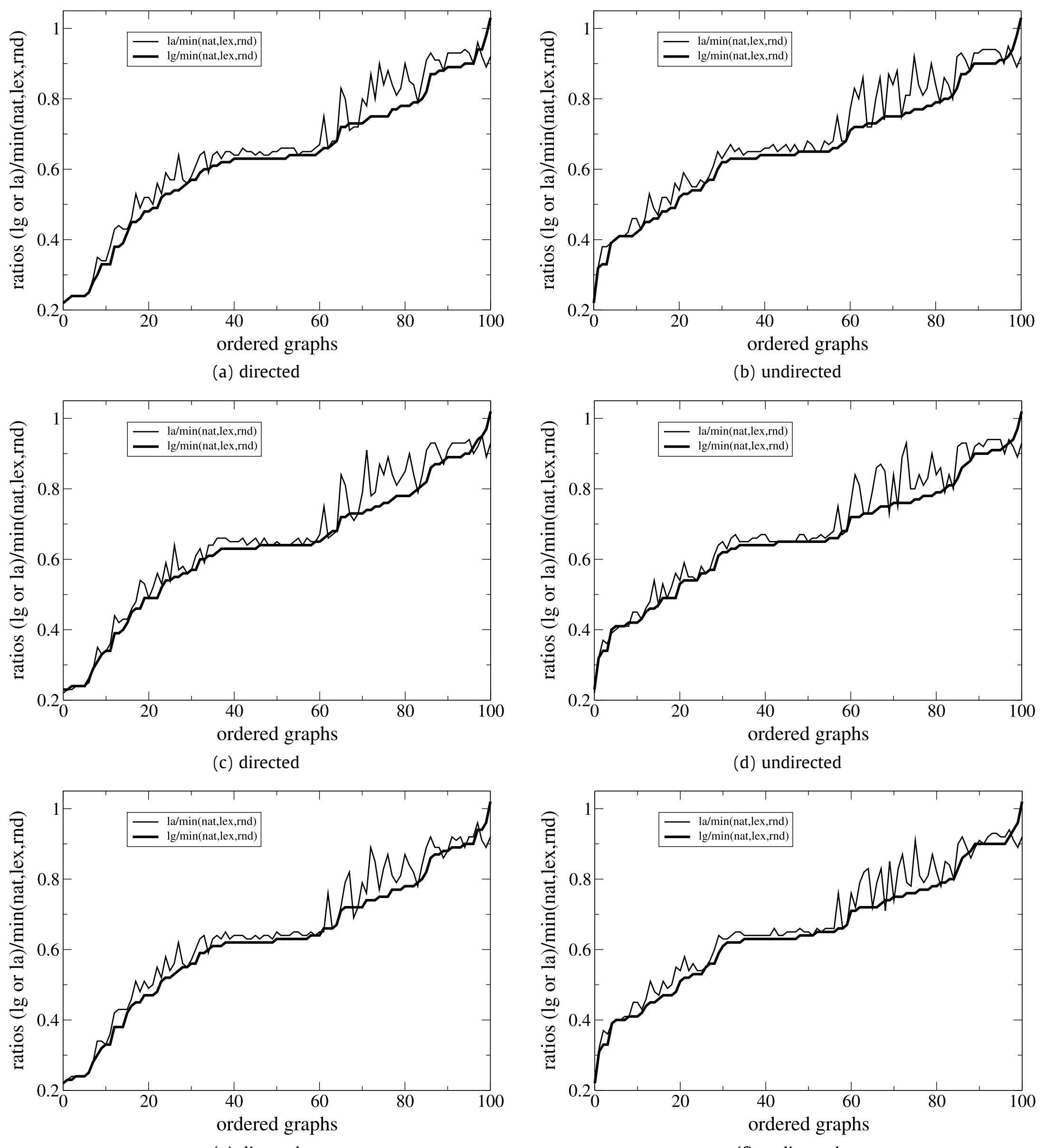
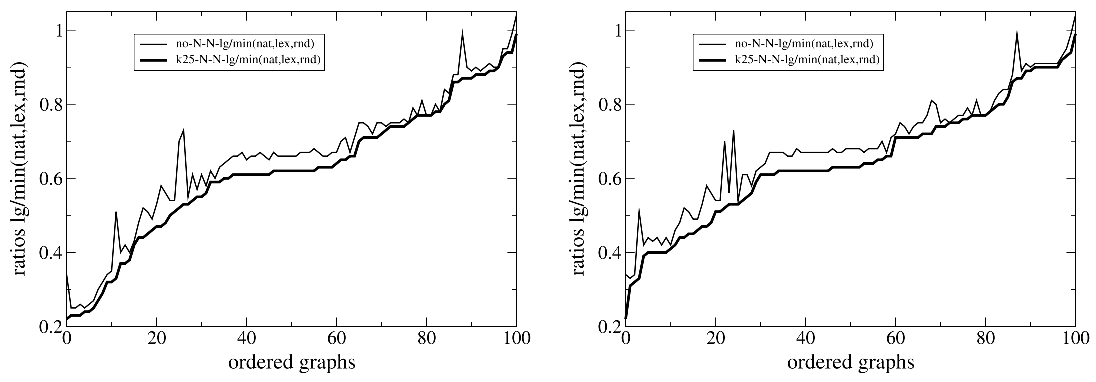


Fig. 3. Comparison results for ms-GMLogA, MinLA + N-N, and $\min(c(G, \pi_{nat}), c(G, \pi_{lex}), c(G, \pi_{rnd}))$. Notations "lg" and "la" correspond to the results of ms-GMLogA and MinLA + N-N, respectively. Three rows correspond to three parameter settings. Row 1: a mild configuration with $k = 5$, $\max(N_i^c) = 1$, number of GS, JOR and compatible relaxations < 20. Row 2: similar to Row 1 with fast algebraic distance coupling, $r = 1$. Row 3: slower configuration with logarithmic increase scale of all parameters during the hierarchy and initial $k = 10$ and $\max(N_i^c) = 2$.

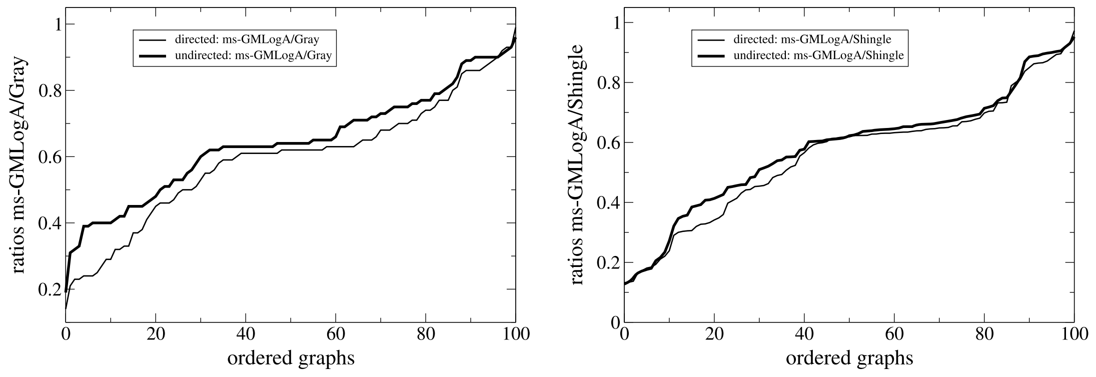
The Gray [6] and double shingle [11] orderings have proven more successful heuristics than lexicographic and natural orderings [22] and thus we present their comparison with ms-GMLogA separately in Fig. 5(a), (b). Two curves depicted in Fig. 5(a) correspond to the comparison of Gray ordering on directed (regular curve) and undirected (bold curve) graphs, respectively. Each point on the curves corresponds to the ratio between β values of ms-GMLogA with default parameters and Gray ordering, respectively. Similarly, the comparison of double shingle ordering [11] is shown in Fig. 5(b). The (double) shingle heuristic is based on the node similarity ordering derived from estimation of Jaccard coefficients. In [11], the authors prove that (double) shingle heuristic will be beneficial for networks that can be described by the preferential attachment model.

Recently, the layered label propagation algorithm (LayeredLPA) was introduced in [1]. This propagation method is based on the Potts model [19]. This algorithm is significantly more successful than natural, random, lexicographic, Gray, and (double) shingle orderings. The success of the label propagation methods is similar to that of the algebraic distance coupling (see Section 3.1.1 and [10]) in which propagation and averaging of random values over the node neighborhoods are employed.



(a) directed (b) undirected

Fig. 4. Comparison of the fastest and the slowest versions of ms-GMLogA. Notations "no-N-N-lg" and "k25-N-N'lg" correspond to the fast and slow versions of ms-GMLogA, respectively.



(a) Gray ordering versus ms-GMLogA (b) Double shingle versus ms-GMLogA

Fig. 5. Comparison of Gray and double shingle orderings versus ms-GMLogA.

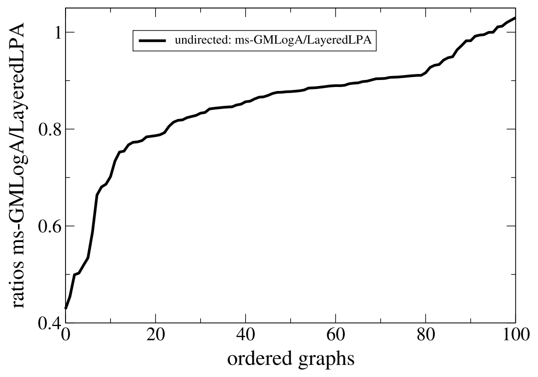


Fig. 6. Comparison of LayeredLPA versus ms-GMLogA.

However, our multiscale method shows better results (see Fig. 6). Note that the LayeredLPA is introduced for undirected graphs only. We believe that introducing the AMG-based framework to the label propagation model can significantly improve its quality.

4.1. Scalability of ms-GMLogA

Good scalability is one of the most important advantages of multiscale algorithms. All components of our scheme are of linear complexity, and the total complexity is $O(|V| + |E|)$. The dependence of the running time on the graph size is depicted in Fig. 7. Each small circle in the figure corresponds to a particular graph. We added to this figure a regression line whose slope is close to 1. The dependence is presented in logarithmic scale. We do not expect a more precise relationship

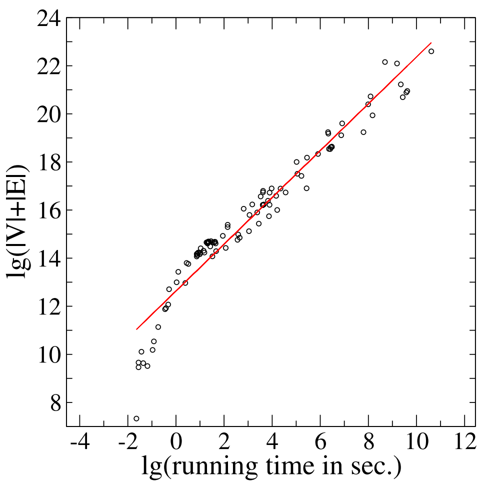


Fig. 7. ms-GMLogA: running time vs graph size.

Table 2
Comparison of spectral method and ms-GMLogA.

Network	Spectral	ms-GMLogA
as-caida20071105	12.4471	6.32
email-Enron	10.3117	7.06
oregon1_010407	11.0662	6.18
p2p-Gnutella06	10.6513	7.13
wiki-Vote	9.2152	7.41

between $|V| + |E|$ and the running time since the implementation is far from being optimized and since it may depend also on structural factors such as the degree distribution of the graph and its diameter.

4.2. Spectral approach

Arrangement of the graph vertices according to the eigenvector (called the Fiedler vector) corresponding to the second smallest eigenvalue of the graph Laplacian is a well-known and successful heuristic used for many ordering problems such as the minimum p-sum and the minimum envelope reduction (see [4,15]). Usually, for large graphs the computation of the Fiedler vector is too expensive, and some approximation is employed [21]. Most of these approximations can be viewed as a global averaging process that works until a particular convergence. As a result, during this process the vertex tends to be located at the weighted average of its neighbors, which makes it suitable for solving the quadratic functionals such as the minimum 2-sum problem [15] but creates a poor solution for the sum of logarithms. Several examples (from [17]) of a comparison between spectral approach and ms-GMLogA are presented in Table 2.

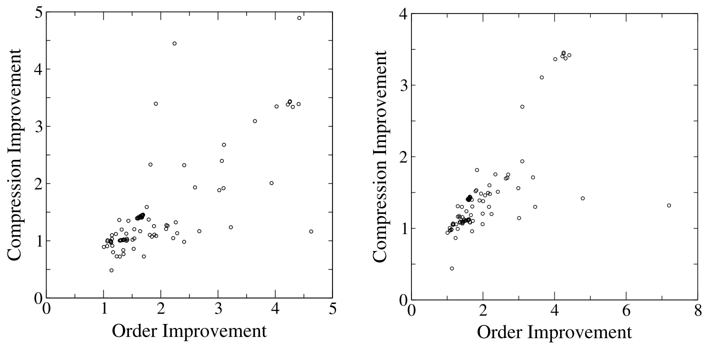
4.3. Compressing the ordered networks

In [11], Chierichetti et al. conjectured that minimizing the unweighted version of MLogA will automatically improve the graph compression. The conjecture has been supported by experimental observations on a few web graphs and social networks. However, for general networks, a better ordering does not always imply a better compression ratio produced by the kind of compressions described in [5]. To illustrate that we present in Fig. 8 a dependence of order improvement on the compression produced in [5].

Given an initial ordering (natural or Gray), we apply our algorithm to obtain a new ordering with smaller β. The x-axis corresponds to the ratio between β values of initial ordering and ms-GMLogA. The y-axis corresponds to the ratio obtained by compressing these orders. On most of the graphs, the conjecture is confirmed. On some graphs, however, we can see the degradation of the compression results. It is observable, in particular, in Fig. 8(b), where the initial arrangement was obtained by Gray ordering that is significantly better than the natural ordering.

5. Conclusions and future work

We have proposed a fast linear algorithm (ms-GMLogA), for compression-friendly graph reordering. The algorithm belongs to the family of multiscale methods [8]. It uses a novel AMG-based coarsening scheme that is reinforced by a modification of algebraic distance couplings [20]. The empirical results on a large benchmark demonstrate its quality and



(a) initial ordering is natural (b) initial ordering is Gray

Fig. 8. Correlation between ordering and compression improvements.

scalability. The model takes into account the edge and node weights that can express different properties of a network model such as known link importance or its access frequency. Overall, we recommend this multiscale framework for practical, compression-friendly network orderings.

As future work we identify the following research directions that have the potential to improve ms-GMLogA:

- development of a learning algorithm to improve the bandwidth h in (8);
- development of a more sophisticated collective refinement of nodes;
- development of a multiscale method for MLinGapA (defined below); and
- more sophisticated multiscale organization of directed graphs.

A more sophisticated collective refinement of nodes can allow interruption of the coarsening earlier. Thus, if some more qualitative approximation can be achieved at the very coarse levels instead of solving exactly the coarsest and applying fast simple refinement at the next few finer level, it can potentially lead to better solutions at the fine levels.

The next natural step is to design first a refinement and then a full multiscale approach for *the minimum logarithmic gap arrangement problem* (MLinGapA) defined in [11] as an alternative model for compression. In this problem the goal is to find a permutation of nodes that minimizes

$$\sum_{i \in V} f_\pi\big(i, out(i)\big), \tag{10}$$

where $f_\pi(i, out(i))$ is a sum of logarithms of gaps between consecutive neighbors of i ordered by π. This problem is also NP-hard [11]; however, we believe that having a suitable refinement and relaxation algorithms would make it possible to adopt the multiscale framework for this problem as well.

The last research direction is related to finding a better multiscale scheme for directed graphs. The presented model constructs an undirected edge ij from a given set of directed edges between i and j by reflecting both directions as an edge weight. In fact, this way of representation is not far from the attempts to solve the problems on nonsymmetric matrix A by applying the known techniques on $A + A'$ or $A \cdot A'$. However, it is known that these methods suffer from several drawbacks. We identify a finding of a more advanced multiscale representation for directed graphs as one of the major future research directions.

Acknowledgements

This work was funded in part by the CSCAPES institute, a DOE project, and in part by DOE Contract DE-AC02-06CH11357. We express our gratitude to Marco Rosa for helping to design experiments with WebGraph package [1].

Appendix A. Parameters

$\Theta_{1,2}$. The threshold parameters Θ_1 and Θ_2 from (4) are responsible for controlling the complexity of the coarse-level problem and the quality of coarsening. It is important to check their robustness when designing a multiscale framework. The presented computational experiments have been executed with $\Theta_1 = \Theta_2 = 1/2$; however, numerical results with other values $0.2 \leqslant \Theta_1, \Theta_2 \leqslant 0.8$ exhibit good behavior as well. Of course, the larger values increase the running time as the coarse graphs become bigger.

ω. Detailed discussion about choosing parameter ω for Jacobi overrelaxation on general graphs is discussed in [10]. In all our computational experiments this parameter was 0.5.

References

[1] Laboratory for Web Algorithmics, http://law.dsi.unimi.it/.
[2] M. Adler, M. Mitzenmacher, Towards compressing web graphs, in: DCC '01: Proceedings of the Data Compression Conference, IEEE Computer Society, Washington, DC, USA, 2001, p. 203.
[3] A. Apostolico, G. Drovandi, Graph compression by BFS, Algorithms 2 (3) (2009) 1031–1044, doi:10.3390/a2031031.
[4] S. Barnard, A. Pothen, H. Simon, A spectral algorithm for envelope reduction of sparse matrices, Numer. Linear Algebra Appl. 2 (4) (1995) 317–334.
[5] P. Boldi, S. Vigna, The Webgraph framework I: compression techniques, in: WWW '04: Proceedings of the 13th International Conference on World Wide Web, ACM, New York, NY, USA, 2004, pp. 595–602, doi:10.1145/988672.988752.
[6] P. Boldi, M. Santini, S. Vigna, Permuting web graphs, in: K. Avrachenkov, D. Donato, N. Litvak (Eds.), WAW, in: Lecture Notes in Computer Science, vol. 5427, Springer, 2009, pp. 116–126.
[7] E.G. Boman, U.V. Catalyurek, C. Chevalier, K.D. Devine, I. Safro, M.M. Wolf, Advances in parallel partitioning, load balancing and matrix ordering for scientific computing, J. Phys. Conf. Ser. 180 (1) (2009) 12008–12013.
[8] A. Brandt, D. Ron, Multigrid solvers and multilevel optimization strategies, in: J. Cong, J.R. Shinnerl (Eds.), Multilevel Optimization and VLSICAD, Kluwer, 2003 (Chapter 1).
[9] A. Brandt, S. McCormick, J. Ruge, Algebraic multigrid (AMG) for automatic multigrid solution with application to geodetic computations, Tech. rep., Institute for Computational Studies, Fort Collins, CO, POB 1852 (1982).
[10] J. Chen, I. Safro, Algebraic distance on graphs, Tech. Rep. ANL/MCS-P1696-1009 (Preprint), Argonne National Laboratory, 2009.
[11] F. Chierichetti, R. Kumar, S. Lattanzi, M. Mitzenmacher, A. Panconesi, P. Raghavan, On compressing social networks, in: KDD '09: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, 2009, pp. 219–228, doi:10.1145/1557019.1557049.
[12] Y. Choi, W. Szpankowski, Compression of graphical structures, in: ISIT'09: Proceedings of the 2009 IEEE Symposium on Information Theory, IEEE Press, Piscataway, NJ, USA, 2009, pp. 364–368.
[13] T. Davis, University of Florida Sparse Matrix Collection, NA Digest 97 (23). URL http://www.cise.ufl.edu/research/sparse/matrices.
[14] J. Díaz, J. Petit, M. Serna, A survey of graph layout problems, ACM Comput. Surv. 34 (3) (2002) 313–356, doi:10.1145/568522.568523.
[15] M. Juvan, B. Mohar, Optimal linear labelings and eigenvalues of graphs, Discrete Appl. Math. 36 (2) (1992) 153–168, doi:10.1016/0166-218X(92)90229-4.
[16] Y. Lai, K. Williams, A survey of solved problems and applications on bandwidth, edgesum, and profile of graphs, J. Graph Theory 31 (1999) 75–94.
[17] J. Leskovec, Stanford Network, Analysis package (SNAP), http://snap.stanford.edu/index.html.
[18] J. Leskovec, L. Backstrom, R. Kumar, A. Tomkins, Microscopic evolution of social networks, in: KDD '08: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, 2008, pp. 462–470, doi:10.1145/1401890.1401948.
[19] R.B. Potts, Some generalized order-disorder transformations, Proc. Cambridge Philos. Soc. 48 (1952) 106–109.
[20] D. Ron, I. Safro, A. Brandt, Relaxation-based coarsening and multiscale graph organization, Tech. Rep. ANL/MCS-P1741-0410 (Preprint), Argonne National Laboratory, 2010.
[21] Y. Saad, Numerical Methods for Large Eigenvalue Problems, Halsted Press, 1992.
[22] I. Safro, B. Temkin, Benchmark for the minimum logarithmic arrangement problem, http://www.mcs.anl.gov/~safro/mloga.html.
[23] I. Safro, D. Ron, A. Brandt, Graph minimum linear arrangement by multilevel weighted edge contractions, J. Algorithms 60 (1) (2006) 24–41.
[24] I. Safro, D. Ron, A. Brandt, Multilevel algorithms for linear ordering problems, J. Exp. Algorithmics 13 (2008), 1.4–1.20.
[25] B.W. Silverman, Density Estimation for Statistics and Data Analysis, Chapman & Hall, London, 1986.
[26] J. Sun, E.M. Bollt, D. Ben-Avraham, Graph compression-save information by exploiting redundancy, J. Statist. Mech. 2008 (06) (2008) P06001.
[27] C. Walshaw, Multilevel refinement for combinatorial optimisation: Boosting metaheuristic performance, in: C. Blum, M. Aguilera, A. Roli, M. Sampels (Eds.), Hybrid Metaheuristic, an Emerging Approach to Optimization, in: Studies in Computational Intelligence, vol. 114, Springer, 2008, pp. 261–289, doi:10.1007/978-3-540-78295-7_9.
[28] H. Zhang, B. Qiu, K. Ivanova, C.L. Giles, H.C. Foley, J. Yen, Locality and attachedness-based temporal social network growth dynamics analysis, J. Amer. Soc. Inform. Sci. Technol. 61 (2009) 964–977.