A Multi-Scale Algorithm for the Linear Arrangement Problem

Yehuda Koren and David Harel

Dept. of Computer Science and Applied Mathematics
The Weizmann Institute of Science, Rehovot, Israel
{yehuda,harel}@wisdom.weizmann.ac.il

Abstract. Finding a linear ordering of the vertices of a graph is a common problem arising in diverse applications. In this paper we present a linear-time algorithm for this problem, based on the multi-scale paradigm. Experimental results are similar to those of the best known approaches, while the running time is significantly better, enabling it to deal with much larger graphs. The paper contains a general multi-scale construction, which may be used for a broader range of ordering problems.

1 Introduction

Many computational tasks involve the combinatorial problem of ordering the vertices of a graph so as to optimize certain objective functions. The reader is referred to [3] for a detailed survey. We concentrate here on a common variant of the problem, known as *the minimum linear arrangement problem (MinLA)*, which consists of placing the n vertices of the graph at positions $1 \ldots n$ on a line, so as to minimize the sum of the edge lengths. Put differently, given n pins, some of which are connected by wires, we want to fix the pins in a linear sequence of holes so as to minimize the total wire length. This variant arises in VLSI design, graph drawing, modeling nervous activity in the cortex and job scheduling; see [3]. As is the case with many ordering problems, MinLA is NP-hard and the corresponding decision problem is NP-complete [4].

In this paper, we devise a linear-time heuristic algorithm for the MinLA problem. Our experiments with it show its attractiveness in terms of output quality and actual running time. The algorithm embodies a novel multi-scale scheme for the MinLA problem, which we have constructed to fit many linear ordering tasks. Hence, it could have applications beyond the MinLA problem.

2 The Minimum Linear Arrangement Problem

Let $G(V, E)$ be a graph, where $V = \{1 \ldots n\}$ is the set of n vertices, and E is the set of weighted edges, with w_{ij} being the weight of edge $\langle i, j \rangle \in E$. Throughout the paper we assume, without loss of generality, that G is connected, otherwise the problem we deal with can be solved independently for each connected component.

Definition 21 *A* linear arrangement *of G is a bijection $\pi : V \longrightarrow \{1, \ldots, n\}$. Equivalently, π is simply a permutation of V. We call $\pi(i)$ the* location *of vertex i.*

Definition 22 *The* cost *of the arrangement π is defined to be:*

$$LA_\pi(G) \stackrel{def}{=} \sum_{\langle i,j\rangle \in E} w_{ij} \cdot |\pi(i) - \pi(j)|$$

The objective of the MinLA problem is to find a permutation π that minimizes $LA_\pi(G)$.

2.1 Algorithms

For certain classes of "well-structured" graphs, such as trees, grids and hyper-cubes, there exist polynomial time algorithms for solving the MinLA problem. However, the general problem is NP-hard. Thus most research involves approximation and heuristic algorithms. Here we briefly describe some relevant algorithms.

Dynamic programming Using dynamic programming, we can find the exact minimum linear arrangement in time $O(2^n \cdot |E|)$ and space $O(2^n)$, which is better than the naive $\Theta(n!)$ algorithm that scans all possible permutations. The complexity of this algorithm makes it useless for all, but very small graphs. However, this algorithm will serve us later on for reordering small subgraphs. The algorithm is based on the fact that the minimum linear arrangement problem possesses a "locality property", to the effect that the best ordering of a set of vertices S that placed to the right of the vertices in the set L and to the left of those in R, is independent of the internal ordering of L and R. This algorithm is described (along with a proof of the locality property) in the full version of this paper [10].

Spectral sequencing Spectral information has been successfully applied to vertex ordering problems, see e.g., [1, 6]. Sequencing the vertices is done by sorting them according to their components in the Fiedler vector, which is the second smallest eigenvalue of the Laplacian matrix associated with the graph. Fortunately, computation of the Fiedler vector can be carried out very rapidly using a multi-scale methods of [2, 9].

Simulated annealing Simulated annealing [8] is a powerful, general (if slow) optimization technique that is appropriate for the MinLA problem. It repeatedly modifies the ordering as follows: Given the current candidate ordering π, a new candidate $\hat{\pi}$ is generated that is close to π. If $LA_{\hat{\pi}} \leqslant LA_\pi$, the new ordering $\hat{\pi}$ is adopted. Otherwise, $\hat{\pi}$ may still be adopted, but with a probability that decreases as the process proceeds.

Petit [11] has conducted extensive tests of many algorithms for the MinLA problem, on several classes of graphs. We quote his conclusion:

> "...the best heuristic to approximate the MinLA problem on sparse graphs is Simulated Annealing when measuring the quality of the solutions. However, this heuristic is extremely slow whereas Spectral Sequencing gives results not too far from those obtained by Simulated Annealing, but in much less time. In the case that a good approximation suffices, Spectral Sequencing would clearly be the method of choice." [11]

Our algorithm, described in the following sections, produces results whose quality is similar to that of simulated annealing, but its running time is much better.

3 The Median Iteration

We now describe the *median iteration*, a rapid randomized algorithm for decreasing the cost of a linear arrangement. The heart of the method is a continuous relaxation of the MinLA problem, where we allow vertices to share the same place, or to be placed on non-integral points. Given a graph $G(V, E)$, a *linear placement* is a function $p : V \longrightarrow \mathbb{R}$, and its *cost* is:

$$LP_p(G) = \sum_{\langle i,j \rangle \in E} w_{ij} \cdot |p(i) - p(j)|$$

Notice that a linear arrangement, which was defined as a bijection $V \longrightarrow \{1, \ldots, n\}$, is a special case of a linear placement.

It is obvious that $LP_p(G)$ is minimal when all the vertices are placed at the same location. Hence, the minimal linear placement problem is not interesting. What will be useful for us is a certain process of minimizing $LP_p(G)$, which we shall use to improve linear arrangements. We begin by defining the median of the places of i's neighbors.

Definition 31 (Median) *Let $G(V, E)$ be a graph and p a linear placement. Given some $i \in V$, the* median *of i's neighborhood is denoted by $med_p^G(i)$.*
Since $med_p^G(i)$ is a usual median, it satisfies:

$$\sum_{j \in N(i), p(j) \leqslant med_p^G(i)} w_{ij} \geqslant \sum_{j \in N(i), p(j) > med_p^G(i)} w_{ij}$$

$$\sum_{j \in N(i), p(j) \geqslant med_p^G(i)} w_{ij} \geqslant \sum_{j \in N(i), p(j) < med_p^G(i)} w_{ij}$$

The main observation is the following:

Proposition 1. *Let $G(V, E)$ be a graph and p a linear placement. Fix the places of all vertices except a single $i \in V$. A location of i that minimizes the linear placement cost is $med_p^G(i)$.*

Proof. See the full version of this paper [10].

We can now define an iterative process for reducing the cost of a linear placement, based on minimizing the cost associated with each vertex separately:

Function Median_Iteration ($G(V, E)$, p, k)
% Parameters:
% $G(V, E)$ - a graph, p - a linear placement, k - no. of sweeps
  repeat k times:
    for every $i \in V$ **do**
      $p(i) \leftarrow med_p^G(i)$

  % When a valid linear arrangement is needed:
  Sort nodes according to respected entries in p
  for every $i \in V$ **do**
    $p(i) \leftarrow \langle$sorted place of $i\rangle$

Suppose that the initial value of p is a valid linear arrangement. When k is overly large, a significant fraction of V will be placed at a single point, leaving us very little information. But, for relatively small k, we will get many small clusters of vertices placed together. This provides important information about the global structure of the linear arrangement. Hence, we can construct a new valid linear arrangement by sorting the nodes according to their values in the linear placement p. The decision about the internal order of vertices that are placed at the same point is random.

We call this method *median iteration*. Since the median can be computed in linear time, the time complexity is $O(k \cdot |E|)$ for the k sweeps, plus $O(n \log n)$ for sorting the nodes by their place. Since we take k to be fixed (a typical value would be $k = 50$), the total time complexity is $O(|E| + n \log n)$. The median iteration is a simple way to reduce the cost of linear arrangement. It addresses the global structure of the ordering, while making local decisions randomly.

4 The Multi-Scale Algorithm

The *multi-scale* (or, *multi-level*) paradigm is a powerful general technique that allows fast exploration of properties related to the 'global structure' of complex objects, that depend on many elements within. Multi-scale algorithms have proved to be successful in a variety of areas in physics, chemistry and engineering. See, e.g., [12]. Multi-scale techniques transform a high-dimensional problem in an iterative fashion into ones of increasingly lower dimensions,via a process called *coarsening*. On the coarsest scale the problem is solved exactly, following which a *refinement* process starts, whereby the solution is progressively projected back into higher and higher dimensions, updated appropriately at each scale, until the original problem is reproduced. A multi-scale algorithm must be tailored for the particular problem at hand, so that the coarsening process keeps the essence of the problem unchanged.

In terms of its use for dealing with graph-theoretic optimization problems, the multi-scale approach is widely used for graph-partitioning, see, e.g., [7]. More recently, Walshaw [13] has used this approach for the TSP and vertex-coloring problems. We have used it for the related problem of drawing graphs aesthetically [5, 9].

4.1 Segment graphs

One of the most prominent requirements of a good multi-scale algorithm is that it keep the inherent structure of the problem unchanged during coarsening. In our case, the form of the MinLA problem, as described in Section 2, will not be preserved during reasonable notions of coarsening, but we can define a more general problem that is preserved during coarsening, and it will contain the original problem as a special case. In fact, this general problem emerges naturally from trying to find an arrangement for a more general entity, that we shall call an *s-graph*, where the *s* stands for *segment*.

Definition 41 *An* s-graph *is a graph $G(V, E)$, whose vertices can be thought of as line segments; a vertex i is associated with a nonnegative real number l_i, denoting its length. Each edge $\langle i, j \rangle$ is associated with two coordinates, ${}_P\langle i, j \rangle$ and $\langle i, j \rangle_P$, denoting the positions of its endpoints inside vertices i and j, respectively. Clearly we have, $0 \leqslant_P \langle i, j \rangle \leqslant l_i$ and $0 \leqslant \langle i, j \rangle_P \leqslant l_j$.*

Fig. 1 shows an example of an s-graph. This is a 3-clique (a triangle), so the vertex set is $\{1, 2, 3\}$. The vertex lengths are $l_1 = 3$, $l_2 = 4$, $l_3 = 2$. Weights of the edges are $w_{13} = 2$, $w_{12} = 4$, $w_{23} = 3$. Finally, the coordinates of the edges are ${}_P\langle 1, 3\rangle = 1$, $\langle 1, 3\rangle_P = 0$, ${}_P\langle 1, 2\rangle = 2$, $\langle 1, 2\rangle_P = 1$, ${}_P\langle 2, 3\rangle = 2$, $\langle 2, 3\rangle_P = 1$.

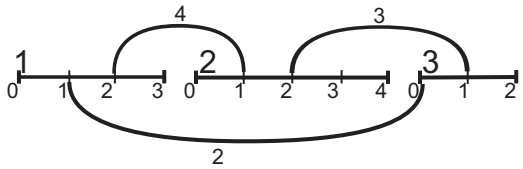

Fig. 1. An s-graph

In a linear arrangement of an s-graph, we place the vertices on a line, in such a way that no two of them intersect. Since we seek an arrangement with short edges, we can safely rule out unused gaps between consecutive vertices. (Though, in the figures we have placed small gaps between consecutive vertices, to make the drawings clearer.) Hence, we define a linear arrangement of an s-graph as follows:

Definition 42 *Let G be an s-graph. A linear arrangement of G is a bijection $\pi : V \longrightarrow \{1, \ldots, n\}$. The location of vertex i is denoted by $p^\pi_G(i)$, and it satisfies $p^\pi_G(i) = \sum_{j,\pi(j)<\pi(i)} l_j$. We will often omit the G in $p^\pi_G(i)$.*

In Fig. 1 we are showing a linear arrangement for which the order of vertices is determined by the bijection: $\pi(1) = 1$, $\pi(2) = 2$, $\pi(3) = 3$. The exact locations of the vertices are: $p^\pi(1) = 0$, $p^\pi(2) = 3$, $p^\pi(3) = 7$.

Definition 43 *Given an s-graph G and a linear arrangement π,* the length of edge *$\langle i, j\rangle$, w.r.t. π, is defined to be:*

$$len^\pi_G(\langle i, j\rangle) \stackrel{def}{=} |p^\pi(j) + \langle i, j\rangle_P - p^\pi(i) -_P \langle i, j\rangle|$$

We will often omit the G in $len^\pi_G(\langle i, j\rangle)$.

Thus, in Fig. 1, $len^\pi(\langle 1, 3\rangle) = 6$, $len^\pi(\langle 1, 2\rangle) = 2$, $len^\pi(\langle 2, 3\rangle) = 3$.

We now generalize the notion of the cost of a linear arrangement (see Def. 22), to handle s-graphs:

Definition 44 *Given an s-graph G, the* cost *of the linear arrangement π is defined to be :*

$$LA_\pi(G) \stackrel{def}{=} \sum_{\langle i,j\rangle \in E} w_{ij} \cdot len^\pi(\langle i, j\rangle)$$

Given a regular graph G (not an s-graph), if we set the length of each vertex to be 1 and for every $\langle i, j\rangle$ we take ${}_P\langle i, j\rangle = \langle i, j\rangle_P = 1$, the definition of the generalized cost of a linear arrangement coincide with the regular case.

4.2 The coarsening process

Let G be an s-graph with n nodes. A single coarsening step would be to replace G with a smaller s-graph G^R, containing only $m < n$ nodes (typically, $m \approx \frac{1}{2}n$). Of course, the structure of G^R should be strongly linked to that of G, so that in some crucial way both will describe approximately the same entity, but on different scales. Moreover, a good linear arrangement of G^R should correspond to a good arrangement of G.

Our attitude to coarsening is to place restrictions on the possible arrangements. These will reduce the search space, so that the restricted problem will be of a "lower dimension", and will involve a smaller graph. We will require that certain pairs of vertices be placed adjacently, in the hope that they will end up being fairly close in the minimal arrangement of G. In this optimistic situation there is a solution in the restricted subspace that is quite close to the optimal solution, up to some local refinement that does not change the global structure of the arrangement but affects only local portions thereof.

Restricting the search space We would like to be able to restrict the search space in a way that satisfies two conflicting goals: First, the degrees of freedom should be significantly reduced, enabling representation by a much smaller graph. Second, the minimal arrangement should be affected only locally, while preserving its global structure.

We have found a way that attempts to achieve these two seemingly competing goals. Consider an initial arrangement π, constructed by some fast algorithm like spectral sequencing or median iteration. For simplicity, assume that the number of vertices n is even. For every pair of vertices v_1, v_2 such that $\pi(v_1) = 2i-1$ and $\pi(v_2) = 2i$ for some $i \geqslant 1$, introduce a restriction that requires any feasible linear arrangement φ to satisfy $\varphi(v_2) = \varphi(v_1) + 1$. The restricted problem is to arrange $n/2$ *restricted vertex pairs*. Hence, the size of the *restricted search space* is $(n/2)!$ while the size of the original search space was $n!$. If n is odd, choose at random some odd k, with $1 \leqslant k \leqslant n$, and restrict consecutive pairs in the series $\pi^{-1}(1), \ldots, \pi^{-1}(k-1), \pi^{-1}(k+1), \ldots, \pi^{-1}(n)$. Vertex $\pi^{-1}(k)$ is not restricted. If the initial arrangement π was not too far from the minimal arrangement, the restrictions just introduced affect the solution only locally, as desired.

Throughout this section, we illustrate the coarsening process using the s-graph G, shown in Fig. 2. All its edges have weight 1. We arbitrarily order the vertices by their names and denote this linear arrangement by π. As the reader can verify, its cost is $LA_\pi(G) = 43$. We should remark that, in general, smarter orderings, such as that produced by the median iteration, work much better than an arbitrary initialization.

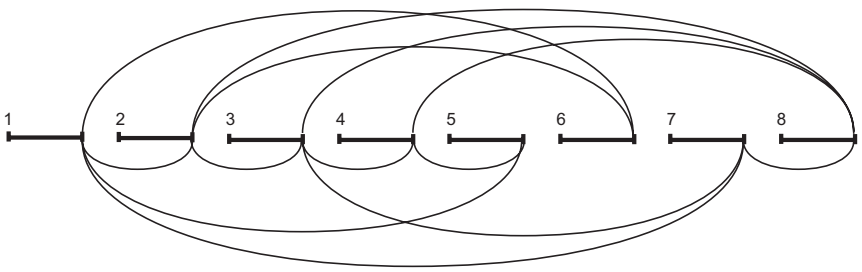

Fig. 2. Linear arrangement of the graph G. Every vertex is of length 1.

As a consequence of the initial linear arrangement π, the set of restricted vertex pairs in the example is $R = \{(1,2),(3,4),(5,6),(7,8)\}$. This restricts every linear arrangement to place vertex 2 immediately after vertex 1, vertex 4 immediately after vertex 3, and so on.

We now show how to formulate the restricted problem as a linear arrangement problem on a much smaller graph.

The coarsening step Given an s-graph $G(V,E)$ and a set $R \subset V \times V$ of restricted vertex pairs, we construct a coarser graph $G^R(V^R,E^R)$, such that there is a 1-1 correspondence between linear arrangements of G^R and linear arrangements in the restricted search space of G. G^R is produced by contracting pairs of restricted vertices. Vertex lengths and edge weights are preserved, by accumulating them. Positions of the new edges are calculated by averaging those of old edges.

Here is the formal construction of G^R. To ease the technicalities, we will be assuming that if $\langle i,j\rangle \notin E$, then $w_{ij} = 0$ and ${}_P\langle i,j\rangle = \langle i,j\rangle_P = 0$. Also, for simplicity, we assume that n is even, so that every $v \in V$ appears in a single restricted pair.

- The coarse vertex set V^R is simply the set of restricted vertex pairs, R.
- The length of a coarse vertex (v_1,v_2) is $l_{(v_1,v_2)} = l_{v_1} + l_{v_2}$
- The coarse edge set is defined as follows:

$$E^R = \left\{ \langle (v_1,v_2),(v_3,v_4)\rangle \;\middle|\; \begin{array}{l} (v_1,v_2),(v_3,v_4) \in V^R, \\ (v_1,v_2) \neq (v_3,v_4), \\ \langle v_1,v_3\rangle \in E \text{ or } \langle v_1,v_4\rangle \in E \\ \text{ or } \langle v_2,v_3\rangle \in E \text{ or } \langle v_1,v_4\rangle \in E \end{array} \right\}$$

- The weight of a coarse edge $\langle (v_1,v_2),(v_3,v_4)\rangle$, denoted as usual by $w_{(v_1,v_2)(v_3,v_4)}$, is $w_{v_1v_3} + w_{v_1v_4} + w_{v_2v_3} + w_{v_2v_4}$.
- The positions of the endpoints of the coarse edge $\langle (v_1,v_2),(v_3,v_4)\rangle$ are the weighted averages of the endpoints of the corresponding fine edges:

$${}_P\langle (v_1,v_2),(v_3,v_4)\rangle = \frac{\left(\sum_{i\in\{v_1,v_2\},j\in\{v_3,v_4\}} w_{ij}\cdot {}_P\langle i,j\rangle\right) + l_{v_1}\cdot(w_{v_2v_3}+w_{v_2v_4})}{w_{v_1v_3}+w_{v_1v_4}+w_{v_2v_3}+w_{v_2v_4}}$$

$$\langle (v_1,v_2),(v_3,v_4)\rangle_P = \frac{\left(\sum_{i\in\{v_1,v_2\},j\in\{v_3,v_4\}} w_{ij}\cdot \langle i,j\rangle_P\right) + l_{v_3}\cdot(w_{v_1v_4}+w_{v_2v_4})}{w_{v_1v_3}+w_{v_1v_4}+w_{v_2v_3}+w_{v_2v_4}}$$

When n is odd, there exists a single unrestricted vertex, v, in which case we add a vertex (v,v) to V^R, where $l_{(v,v)} = l_v$. Definitions regarding the edges adjacent to (v,v) should be slightly modified.

Hence, for our example graph G, as arranged in Fig. 2, we will get the coarse graph G^R shown in Fig 3. We denote the linear arrangement of G^R by φ. The reader can verify that its cost is $LA_\varphi(G^R) = 40$.

There is a simple 1-1 correspondence between the restricted linear arrangements of the fine graph G and the linear arrangements of the coarse graph G^R:

Definition 45 (Correspondence) *Let G be a graph, R a set of restricted vertex pairs, and π a restricted linear arrangement of G. Now let φ be a linear arrangement of the*

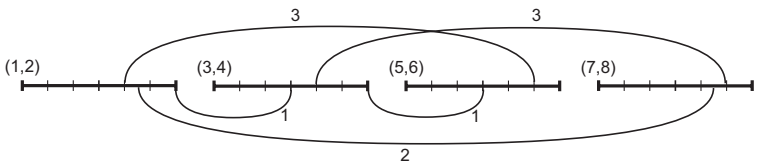

Fig. 3. The graph G^R that is obtained by coarsening the graph of Fig. 3. The length of each vertex is 2, and weights and coordinates of edges were calculated so as to preserve the information of the original graph.

respective coarse graph G^R. The two arrangements are corresponding *if for every $(v_1, v_2) \in R$, we have:*

$$\pi(v_1) = 1 + \sum_{\varphi((i,j)) < \varphi((v_1,v_2))} |(i,j)|$$

where

$$|(i,j)| \stackrel{def}{=} \begin{cases} 2, & i \neq j \\ 1, & i = j \end{cases}$$

Equivalently, in the inverse direction, for every $(v_1, v_2) \in R$:

$$\varphi((v_1,v_2)) = \sum_{(i,j)\in R,\, \pi(i) \leqslant \pi(v_1)} 1$$

Notice that when n is even $(i,j) \in R$ implies that $i \neq j$, so we can simply write the condition for correspondence as follows: for every $(v_1, v_2) \in R$:

$$\pi(v_1) = 2 * \varphi((v_1,v_2)) - 1$$

Using the close relationship between lengths of vertices in G and G^R, we observe that for two corresponding linear arrangements π and φ, the actual locations of the vertices are related, for each $(v_1, v_2) \in R$, by:

$$p^{\pi}_{G}(v_1) = p^{\varphi}_{G^R}((v_1,v_2)) \tag{1}$$

It is straightforward to verify in our example that π and φ, are corresponding linear arrangements. Notice that: $p^{\pi}_{G}(1) = p^{\varphi}_{G^R}((1,2)) = 0$, $p^{\pi}_{G}(3) = p^{\varphi}_{G^R}((3,4)) = 2$, $p^{\pi}_{G}(5) = p^{\varphi}_{G^R}((5,6)) = 4$, $p^{\pi}_{G}(7) = p^{\varphi}_{G^R}((7,8)) = 6$.

During coarsening, several edges become self-loops, and are canceled. We can calculate the cost related to these lost edges:

Definition 46 *Given a graph $G(V,E)$ and a set of restricted vertex pairs R, define* the internal cost of *R to be:*

$$C(R) = \sum_{\langle i,j\rangle \in E, (i,j)\in R} w_{ij} \cdot (\langle i,j\rangle_P + l_i -_P \langle i,j\rangle)$$

The most important fact is that the costs of two corresponding linear arrangements are identical up to adding $C(R)$, which is independent of the particular arrangements.

Lemma 1. *Let G be a graph and R a set of restricted vertex pairs. Let π and φ be corresponding linear arrangements of G and G^R, respectively. Then, $LA_\pi(G) = LA_\varphi(G^R) + C(R)$.*

The proof is rather technical and is given in the full version of this paper [10].

In the example graph three edges lie within a restricted pair, and they are $\langle 1, 2\rangle$, $\langle 3, 4\rangle$, $\langle 7, 8\rangle$. Thus, the internal cost is $C(R) = 3$. In agreement with Lemma 1, $LA_\pi(G) = LA_\varphi(G^R) + C(R) = 40 + 3$.

As can be seen in Fig. 3, the visual complexity of G^R is much lower than that of G. In fact, it is not too hard to compute the minimum linear arrangement φ^* of G^R, shown in Fig. 4(a). Its cost is $LA_{\varphi^*}(G^R) = 24$. After solving the problem for the coarse graph, we interpolate the resulting arrangement to the original graph, obtaining the corresponding linear arrangement π^* of G, as shown in Fig. 4(b). In accordance with Lemma 1, $LA_{\pi^*}(G) = LA_{\varphi^*}(G^R) + C(R) = 24 + 3$, a significant decrease in the arrangement cost, which may also be appreciated visually.

We should remark that in the more general case the coarse graph would still be too large to facilitate finding its optimal linear arrangement. Hence, as we shall explain, our strategy is to continue recursively with the coarsening process until we reach a reasonably small graph.

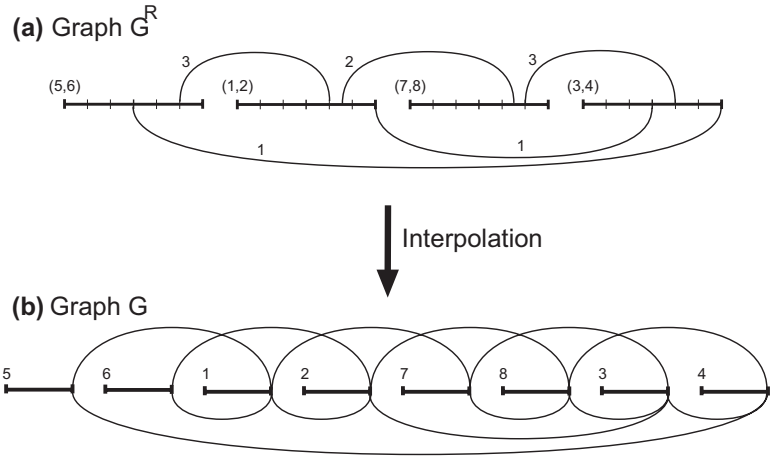

Fig. 4. Optimizing the linear arrangement of G^R facilitates the construction of a better arrangement for G.

We conclude that given a set R of restricted vertex pairs, we can construct a coarser graph G^R with $\lceil \frac{n}{2} \rceil$ vertices. Linear arrangements of G^R correspond to linear arrangements in the restricted search space of G, and have the same costs (up to adding a uniform constant). For reasonable restrictions, the restricted search space contains arrangements that have structure similar to those of the minimum cost arrangements. Hence, we may hope that a good arrangement of G^R corresponds to a reasonably good arrangement of G, and this arrangement can become a truly good one by the local refinement process we now describe.

4.3 Local refinement

Given a linear arrangement π of a graph G, we seek a method that improves the quality of π. Now, in our approach π will typically possess a satisfactory global structure, as a result of minimizing the problem on the restricted search space. Consequently, we have constructed a local refinement process that specializes in locally optimizing arrangements. The idea is to take each k consecutive vertices in π and to rearrange them internally such that the global arrangement cost is minimized. Such an optimal arrangement is achieved using the dynamic programming algorithm mentioned in Subsection 2.1, in a time $O(2^k)$. To improve the results the process can be repeated several times.

The local refinement process is depicted in Figure 5. Taking k to be constant (in our experiments we usually set $k = 6$), the time complexity is $O(|E|)$, and the space requirements are also linear in the size of the input.

Function Local_Refine ($G(V = \{1, \ldots, n\}, E), \pi$)
% Parameters:
% $G(V, E)$ – an s-graph , π – a linear arrangement of V
% Constants:
% $interval[= 6]$ – number of consecutive vertices to optimize together
% $repetitions[= 5]$ – number of iterations of the algorithm
% Auxiliary function:
% MinLA_local($G(V, E)$, π, j, $j + k - 1$) – finds best internal
% ordering of the k vertices placed in $\pi[j], \ldots, \pi[j + k - 1]$.
for $i = 1$ to $repetitions$
for $j = 1$ to $n - interval + 1$
MinLA_local($G(V, E)$, π, j, $j + interval - 1$)
end for
end for

Fig. 5. The local refinement process

4.4 Putting it all together

We can now put all this together to obtain the multi scale algorithm.

Preprocessing stage Prior to running the algorithm we want to obtain, rapidly, a reasonable linear arrangement. To that end, we first order the vertices using spectral sequencing (see Section 2) and then improve the result by applying the median iteration.

The V-cycle Our basic multi-scale tool is the *V-cycle*, which starts by refining the arrangement locally. The intention of the refinement is not only to minimize the arrangement cost, but to also improve the quality of the coarsening step that follows it. The next step is to coarsen the graph based on restricting consecutive vertex pairs of the current arrangement. The problem is then solved in the restricted solution space, by running the V-cycle recursively on the coarse graph. Once we have found a good solution in the restricted solution space, we refine it locally (in the full solution space).

In the most optimistic case, the optimal solution on each scale is reachable from the best restricted solution by local optimization, so that we are guaranteed to find it. Our

hope is that in a realistic case a pretty good solution is reachable in this fashion, which, of course, depends on the quality of the refinement process and on the restrictions we impose.

Figure 6(a) shows the V-cycle algorithm. It uses several functions. The first of these, 'coarsen(G, π, G^R, π^R)', receives an s-graph G and a linear arrangement of its vertices, π, and produces a coarse graph G^R and a linear arrangement its vertices, π^R. Here, π and π^R are corresponding linear arrangements as per Def. 45. The way to construct G^R and π^R is described in Subsection 4.2. The function 'interpolate(G^R, π^R, π)', receives a coarse graph G^R and a linear arrangement π^R, and produces the corresponding linear arrangement of the fine graph, π.

The recursion in the V-cycle continues as long as G is 'large enough', and a good termination condition would be when the graph is small enough to facilitate finding the optimum linear arrangement directly. In all the graphs that we have tested, local refinement was useless after more than five levels of recursion, due to the good global structure of the initial arrangement.

All the functions in the V-cycle run in linear time and space, and the recursive call is to a problem of approximately half the size. Thus, the time and space complexity of the V-cycle algorithm is $O(|E|)$.

(a)

Function V-cycle ($G(V, E)$, π)
% Parameters:
% $G(V, E)$ – an s-graph,
π – a linear arrangement of V

Local_Refine(G, π)
if G is 'large enough' **then**
coarsen(G, π, G^R, π^R)
V-cycle(G^R, π^R)
interpolate(G^R, π^R, π)
Local_Refine(G, π)
end if

(b)

Function MS_MinLA ($G(V, E)$, π, k)
% Parameters:
% $G(V, E)$ – an s-graph
% π – a linear arrangement of V
% k – no. of V-cycle iterations

Spectral_Sequencing(G, π)
Median_Iteration(G, π, 40)
for $i = 1$ to k
Median_Iteration(G, π, 10)
V-cycle(G, π)
end for

Fig. 6. (a) The V-cycle (multi-scale) algorithm **(b)** The full algorithm

Iterating the V-cycle The V-cycle can benefit from starting out with a better arrangement. Thus, repeating the V-cycle can improve its results. In fact, this kind of iteration is common in multi-scale algorithms; see [12]. Iterating the V-cycle can only decrease the cost of the arrangement. In fact, our experience with the algorithm is that after a few iterations the process seems to converge, after which improvement is very slow, if at all. We can obtain better results by carrying out several median iterations (e.g., 10), before entering each new V-cycle. This perturbs the arrangement and provides the next V-cycle with a different starting point, thus overcoming premature convergence. In practice we gradually reduce the number of sweeps of the median iteration, lessening its effect; see [10]. The full algorithm appears in Figure 6(b).

5 Experiments

A case study We want to demonstrate the benefit of embedding the local refinement process inside the multi-scale scheme. To that end, we have chosen the 10-dimensional hyper-cube, which consist of 1024 vertices and 5120 edges. What we have observed for this example is typical of many graphs we have considered. The optimal linear arrangement of this graph, π^*, can be computed efficiently, and its cost is 523776; see [11].

As the first stage we applied spectral sequencing to the hyper-cube, resulting in an arrangement with cost 659490. We then improved the arrangement using 40 median iterations. The cost of the resulting arrangement, π^0, is 599958. We should note that this is the random part of the experiment, and other runs provide quite different arrangements (while the median iteration consistently improves the initial arrangement rather significantly).

We then tried to improve π^0 in two ways. First, we applied 100 iterations of the Local_Refine function (setting the local constant $repetitions$ to 100), and obtained an arrangement with cost 593120. More iterations of Local_Refine did not decrease the cost any further. Second, we applied a single V-cycle to π^0. During this, the Local_Refine function performed only 5 iterations in each of its executions, thus the running time was much faster. The result of this was the optimal linear arrangement π^*, a clear improvement.

This illustrates the effect of embedding the local refinement process in the V-cycle multi-scale scheme,which improves both running time and output quality. In general, embedding a local refinement process in a multi-scale scheme adds global considerations to the refinement process, because local moves on a coarse graph express more global ones on the original graph. Thus, in some sense, the "wisdom" of a multi-scale algorithm can be divided into two parts: One is related to local optimization and it resides in the local optimization process, and the other deals with the global properties of the problem and it resides in the coarsening construction.

Petit's test suite We have implemented the algorithm using C++, and use our ACE algorithm [9] for spectral sequencing. The program runs on a 700MHZ Pentium III, under Windows NT. We have tested it on a test suite of graphs compiled by Petit [11], chosen so as to represent several graph families.

Petit computed linear arrangements of these graphs using many algorithms. The best results were obtained by first running spectral sequencing, and then carrying out a refinement using simulated annealing (SA). The SA process was run for $C \cdot n^2$ iterations ($C > 1$ is a constant related to the rate of temperature decrease in SA). We cannot compare his running times with our directly, since the platforms are different. But to get an impression, the running time of SA for the largest graph in Petit's set, the whitaker3, was over 11 hours. We provide the costs computed by SA in Table 1.

For each graph in this set, we ran our multi-scale algorithm (that of Figure 6(b)) first with a single V-cycle and then with ten (i.e., with $k = 1$ and $k = 10$). The results appear in Table 1, which also provides the results of spectral sequencing and median iteration — the first stage of the algorithm. The quality of our results after 10 V-cycle iterations is comparable to that of Petit's SA, but our running time is significantly better. The results may be further improved by executing more iterations and/or by increasing

the value of the constant $interval$ in the function Local_Refine. However, the rate of improvement would be slow.

Graph Name	Size |V|	|E|	Spectral Sequencing	Median Iteration	Multi-Scale 1 Iteration	Multi-Scale 10 Iterations	Cost using SA
randomA1	1000	4974	1156890/.08	1020028/.03	938168/2.39	909126/22.94	900992
randomA2	000	24738	7377237/.27	7284497/.42	6755035/7.02	6606174/63.94	6584658
randomA3	1000	49820	15279645/.38	16543660/.96	14731040/12.12	14457452/109.17	14310861
randomA4	1000	8177	2167121/.11	1955837/.05	1807038/3.15	1765217/30.04	1753265
randomG4	1000	8173	195054/.06	175879/.06	154990/2.11	149513/18.97	150940
hc10	1024	5120	580910/.02	542476/.03	523776/1.9	523776/18.51	548352
mesh33x33	1089	2112	35750/.04	34118/.01	32486/1.04	31729/9.91	34515
bintree10	1023	1022	52992/.09	6114/0	4246/.88	3950/7.96	4069
3elt	4720	13722	429086/.43	394238/.11	385572/6.55	373464/61.24	375387
airfoil1	4253	12289	352897/.37	312387/.11	305191/6.53	291794/61.02	288977
whitaker3	9800	28989	1259607/.92	1238557/.28	1226902/13.86	1205919/127.79	1199777
c1y	828	1749	103224/.02	71359/.01	66836/.92	64934/8.88	63858
c2y	980	2102	95346/.03	84259/.01	82070/1.12	80148/10.81	79500
c3y	1327	2844	175700/.04	145332/.02	137131/1.58	127315/15.3	124708
c4y	1366	2915	133044/.05	124576/.02	121460/1.62	118437/15.57	117254
c5y	1202	2557	144603/.04	115239/0.02	109280/1.39	104076/13.33	102769
gd95c	62	144	599/.02	595/0	509/.08	509/.61	509
gd96a	1076	1676	170700/.04	122567/.01	115525/1.24	106668/11.94	104698
gd96b	111	193	1836/0	1825/.01	1435/.11	1434/.98	1416
gd96c	65	125	701/0	601/0	522/.06	519/.6	519
gd96d	180	228	3691/0	2807/.01	2438/.16	2420/1.53	2393

Table 1. The test suit of Petit [11]. For each graph the table shows the cost of the linear arrangement and the running time in seconds (separated by a slash), for spectral sequencing, median iteration, multi-scale with a single V-cycle and multi-scale with 10 iterated V-cycles. The times for multi-scale include spectral sequencing, median iteration and V-cycles. The rightmost column shows the cost computed by Petit using simulated annealing (SA).

6 Discussion

We have presented a multi-scale algorithm for the minimum linear arrangement problem. It produces arrangements on a par with the best known algorithms, but requires significantly less time, allowing it to deal with much larger graphs (as shown in [10]).

The heart of the algorithm is a novel construction of a hierarchy of coarse graphs, corresponding to increasingly restricted parts of the original problem. A local refinement scheme becomes considerably improved when embedded in the multi-scale construction. In fact, the multi-scale construction is independent of the specific refinement heuristic, and we believe that other heuristics could benefit from being embedded in such a multi-scale scheme. Also, variants of the proposed multi-scale construction can be used for other vertex ordering problems, such as bandwidth or cutwidth minimization [3].

An interesting property of our multi-scale construction has to do with the way we build coarse graphs. The common approach to coarsening is based solely on the graph 's structure. Most frequently such a coarsening is performed by contracting a set of edges, such as those that participate in a maximal matching (see, e.g. [7]). In contrast, our approach to coarsening relies on a specific solution of the problem, on which we impose several restrictions. When we have a globally good approximate solution, our approach has the advantage of utilizing the wisdom of this solution for peforming a coarsening that forces reasonable restrictions. On the other hand, coarsening based on graph struc-

ture is most often very local in nature, and may impose globally bad restrictions, like identifying vertices that should be very distant in the optimal solution. See also [13].

The median iteration process we have proposed seems to have value in its own right. It is actually an extremely fast method for decreasing the cost of an arrangement, using a continuous relaxation of the original problem. In [10] it was applied successfully to graphs with millions of edges.

Acknowledgement

We would like to thank Chris Walshaw for his useful comments and corrections.

References

1. J. E. Atkins, E. G. Boman and B. Hendrickson, "A Spectral Algorithm for Seriation and the Consecutive Ones Problem", *SIAM Journal on Computing* **28** (1998), 297–310.
2. S. T. Barnard and H. D. Simon, "A Fast Multilevel Implementation of Recursive Spectral Bisection for Partitioning Unstructured Problems", *Concurrency: Practice & Experience* **6** (1994), 101–117.
3. J. Diaz, J. Petit and M. Serna, "A Survey on Graph Layout Problems", Technical report LSI-00-61-R, Universitat Politècnica de Catalunya, Departament de Llenguatges i Sistemes Informàtics, 2000.
4. M. R. Garey and D. S. Johnson, *Computers and Intractability: a Guide to the Theory of NP-Completeness*, W. H. Freeman, 1979.
5. D. Harel and Y. Koren, "A Fast Multi-Scale Method for Drawing Graphs Nicely", *Proceedings of Graph Drawing'00* , Lecture Notes in Computer Science, Vol. 1984, Springer Verlag, pp. 183–196, 2000.
6. M. Juvan and B. Mohar, "Optimal Linear Labelings and Eigenvalues of Graphs", *Discrete Applied Math.* **36** (1992), 153–168.
7. G. Karypis and V. Kumar, "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs", *SIAM Journal on Scientific Computing* **20** (1999), 359–392.
8. S. Kirkpatrick J. Gelatt and M.P. Vecchi, "Optimization by Simulated Annealing", *Science* **220** (1983), 671-680.
9. Y. Koren, L. Carmel and D. Harel "ACE: A Fast Multiscale Eigenvectors Computation for Drawing Huge Graphs", Technical Report MCS01-17, Faculty of Mathematics and Computer Science, The Weizmann Institute of Science, 2001. To appear in Proc. IEEE Information Visualization 2002.
10. Y. Koren and D. Harel "Multi-Scale Algorithm for the Linear Arrangement Problem", Technical Report MCS02-04, Faculty of Mathematics and Computer Science, The Weizmann Institute of Science, 2002. Available at: `www.wisdom.weizmann.ac.il/reports.html`
11. J. Petit, "Experiments on the Minimum Linear Arrangement Problem", Technical report LSI-01-7-R, Universitat Politècnica de Catalunya, Departament de Llenguatges i Sistemes Informàtics, 2001.
12. S. H. Teng, "Coarsening, Sampling, and Smoothing: Elements of the Multilevel Method", *Algorithms for Parallel Processing*, IMA Volumes in Mathematics and its Applications, Vol. 105, Springer Verlag, pp. 247–276, 1999.
13. C. Walshaw, "Multilevel Refinement for Combinatorial Optimisation Problems", Technical Report 01/IM/73, Comp. Math. Sci., Univ. Greenwich, London, UK, 2001.